VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Microsoft Robotics Studio - použití jazyka VPL

Microsoft Robotics Studio - Using VPL Language

2010

Petr Madecki

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2010 .....

Na tomto místě bych rád poděkoval paní RNDr. Elišce Ochodkové, Ph.D. za pomoc nejen při návrhu programů pro robota, ale také za řadu podnětných návrhů pro psaní této diplomové práce.

Abstrakt

Tato diplomová práce se zabývá problematikou Microsoft Robotics Developer Studia, konkrétně praktické využití vizuálního programovacího jazyka (VPL). V tomto jazyce bylo vytvořeno několik aplikací pro demonstraci funkčnosti robota LEGO MINDSTORMS NXT 2.0. Cílem práce je popsat jazyk VPL, navrhnout, naimplementovat a otestovat ukázkové aplikace pro robota NXT.

Klíčová slova: MRDS, VPL, robot, LEGO MINDSTORMS NXT

Abstract

This diploma paper is about Microsoft Robotics Developer Studio, specifically the practical use of Visual Programming Language (VPL). Several applications were created in this language for demonstration of functions of robot LEGO MINDSTORMS NXT 2.0. The aim is to describe language VPL, design, implement and test sample application for robot NXT.

Keywords: MRDS, VPL, robot, LEGO MINDSTORMS NXT

Seznam použitých zkratek a symbolů

CCR	–	Concurrency and Coordination Runtime
COM	–	Component Object Model
DSS	–	Decentralized Software Services
DSSCP	–	Decentralized Software Services Command Prompt
DSSME	–	Decentralized Software Services Manifest Editor
DSSP	–	Decentralized Software Services Protocol
MRDS	–	Microsoft Robotics Developer Studio
MVS	–	Microsoft Visual Studio
RGB	–	Red Green Blue
URI	–	Uniform Resource Identifier
USB	–	Universal Serial Bus
VPL	–	Visual Programming Language
VSE	–	Visual Simulation Environment
WWW	–	World Wide Web
XML	–	eXtensible Markup Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V dnešním moderním světě se čím dál více setkáváme s pojmy automatizace, či robotizace. Právě zmíněná robotizace se za posledních pár let stala velice rozšířeným pojem. Roboti se využívají jak v lékařství, automobilovém průmyslu, podmořském výzkumu, záchranných pracích, domácnosti nebo při výuce na střední škole. Právě k poslední možnosti využití robotů je směřována i tato práce. Cílem práce je popsat jazyk VPL, navrhnout, naimplementovat a otestovat ukázkové aplikace pro robota NXT. Postupně se seznámíme s problematikou vývojového prostředí MRDS a jeho vizuálním programovacím jazykem VPL. Bude zmíněno okrajově i vývojové prostředí NXT-G, které je vyvíjeno přímo pro použitého robota. Formou jednotlivých ukázkových aplikací postupně ověříme funkčnost všech senzorů, motorů a displaye.

V kapitole 2, *Microsoft Robotics Developer Studio*, je popsáno vývojové prostředí MRDS a jeho architektura s výjimkou VPL (tomu je věnována následující 3 kapitola).

Ve 3. kapitole, *Visual Programming Language*, je představen jazyk VPL, jaké možnosti nabízí a kdo jej může využívat. Je zde ukázka vytvoření prvního programu v tomto prostředí, jak se vytváří aktivity a jejich propojení. Jak se spouští vytvořený program a co vše se k němu vytváří a vypisuje při spuštění. Na závěr kapitoly je zmíněno, jak se dá převést vytvořený program (diagram) na službu spolu s ukázkou manifestu a kouskem zdrojového kódu.

Kapitola 4, *LEGO MINDSTORMS NXT 2.0*, představí stavebnici LEGO MINDSTORMS NXT ve verzi 2.0. Okrajově je zmíněn rozdíl mezi předchozí a aktuální verzí. Dále následuje popis prostředí NXT-G (prostředí vytvářené přímo pro LEGO roboty), jednotlivých senzorů, servomotoru a NXT kostky.

Kapitola 5, *Výčet a specifikace řešených dílčích úloh*, se na začátku věnuje popisu použitého hardwaru a softwaru pro tuto práci a následně představí deset programů, které pro ni byly vytvořeny.

2 Microsoft Robotics Developer Studio

Microsoft Robotics Developer Studio (MRDS) je platforma vyvinutá společností Microsoft pro tvorbu algoritmů nebo programů pro řízení robotických aplikací a jejich realtimovou simulaci v trojrozměrném simulačním prostředí. Vývoj algoritmů je možný pro širokou škálu různých platforem a hardware. Důležitá vlastnost je, že se dají přidávat vlastní části rozhraní pro vývoj, řízení a simulaci specifického hardware.

Prostředí MRDS je distribuováno ve třech verzích. Academic verze pro akademické použití šířená přes MSDNAA a DreamSpark. Placená verze Standard pro komerční použití v ceně 10 000 Kč. Verze Expres je dostupná ke stažení z internetových stránek MRDS pro použití zdarma, neobsahuje však nástroje jako verze Academic/Standard.

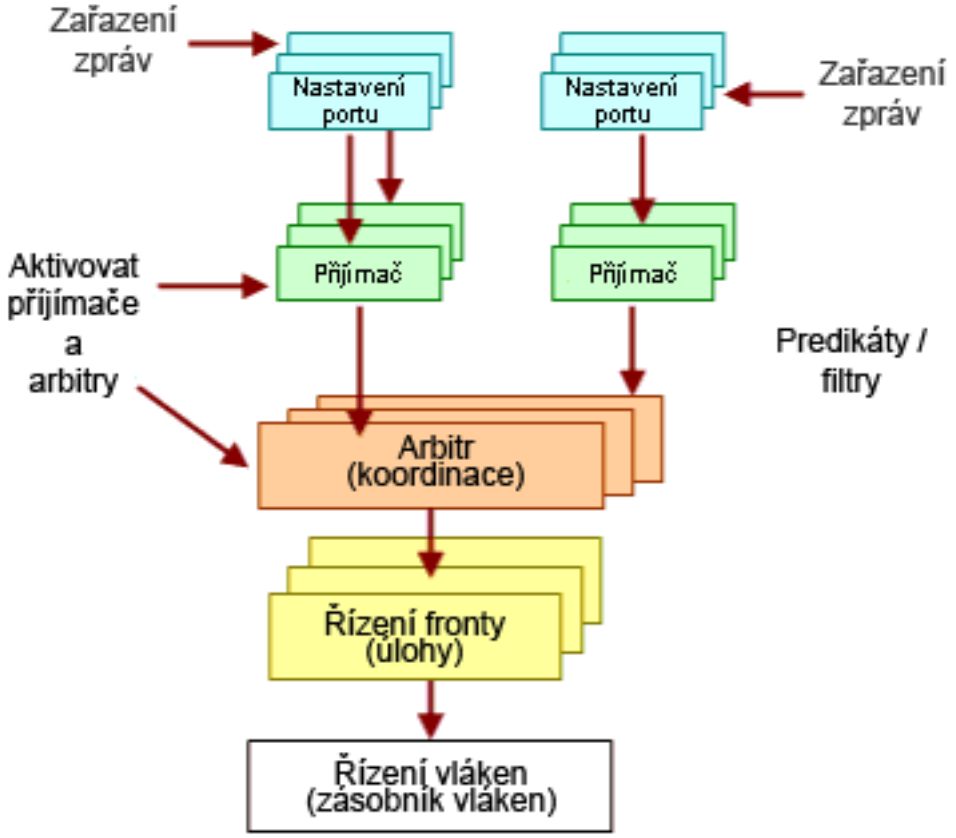
Pro snadné programování a velkou variabilitu vývoje v prostředí MRDS je možné vyvíjet robotické algoritmy v několika programovacích jazycích. Stěžejním programovacím jazykem je C# (Csharp) nebo Visual Basic.Net (VB.NET), podpora je ale i pro další jazyky, jako je JScript, Microsoft IronPython a C++. Prostředí MRDS poskytuje i možnost implementovat rozšiřující rozhraní pro další jazyky. Microsoft žádné další rozhraní nenabízí a proto by si uživatel musel rozhraní sám naprogramovat.

Pro vývojáře, kteří neovládají žádný programovací jazyk, je v prostředí MRDS implementována podpora Microsoft Visual Programmind Language (VPL). Je to vizuální programovací jazyk, ve kterém se programuje pomocí systému chyť a táhni (drag and drop). Pro programátory, kteří programují v pokročilém programovacím jazyce, je v prostředí MRDS podpora pro programování v Microsoft Visual Studiu (MVS), nebo v jiném programovacím prostředí, které podporuje jazyky MRDS. Například to může být NetBeans nebo Eclipse. Výhoda používání MVS je v tom, že jsou pro MVS připraveny šablony pro nové projekty a návody, jak programovat.

MRDS nabízí integrované prostředí založené na platformě .NET. Prostředí MRDS obsahuje množství knihoven, které obstarávají funkce pro robotické aplikace.

2.1 Concurrency and Coordination Runtime

Hlavní knihovna je Concurrency and Coordination Runtime (CCR), která obstarává koordinaci mezi činnostmi robotů, jako jsou například pohyb, informace ze snímačů, reakce na informace a další činnosti podle vlastností robota. Je to vlastně multi-taskingový (více-úlohový) správce zodpovědný za koordinaci více-úlohových aplikací pro MRDS. Zjednodušeně řečeno, spravuje vlákna a zajišťuje správné sladění a součinnost. Architektura CCR je uvedena na obrázku 1.



Obrázek 1: Architektura CCR [4]

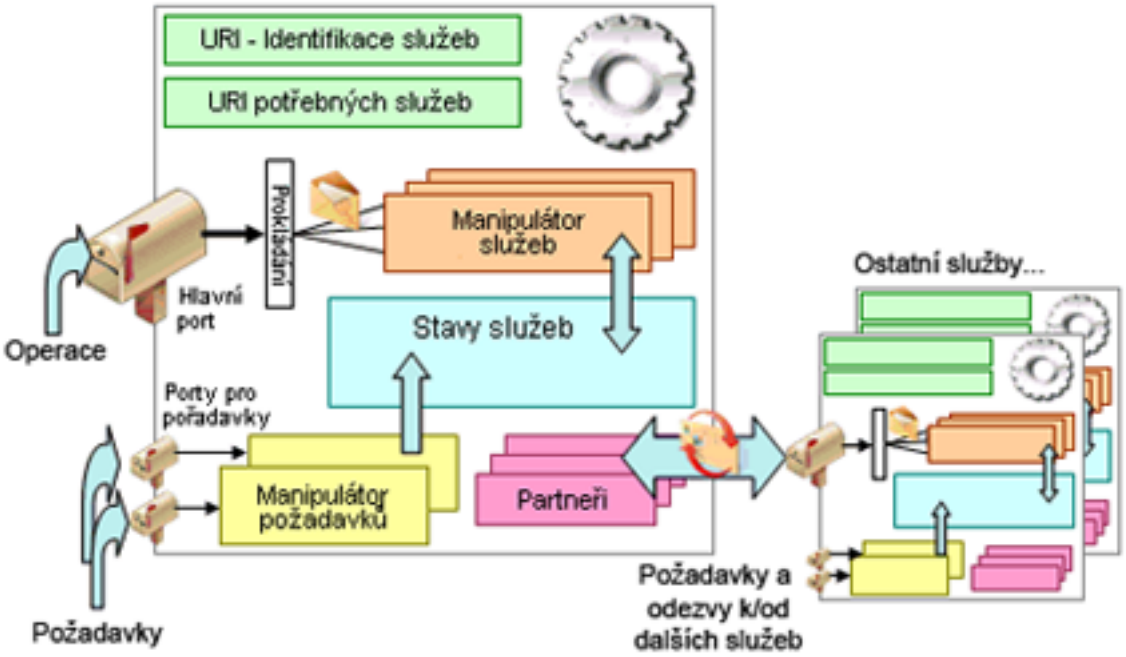
Když přijde příkaz (zpráva) od řídící aplikace přes nastavený port, aktivuje se přijímač a arbitr. Přijímač přijme zprávu, zjistí její druh a pošle zprávu arbitrovi, který ji rozřadí do fronty zpráv, podle druhu a priority zprávy. Zprávy spustí úlohu ve vytvořeném vlákně, které je řízeno a spravováno v zásobníku vláken.

Služby jsou obsažené v knihovnách a poskytují přístup k naprogramovaným funkcím. Tyto funkce se používají pro komunikaci s hardwarem, provádí výpočty pokročilých algoritmů, vytváření simulačního prostředí, řízení motorů, čtení dat ze senzorů a další robotické funkce.

2.2 Decentralized Software Services

Další důležitou knihovnou je Decentralized Software Services (DSS), která je grafickou nadstavbou pro CCR. Vytváří grafické prostředí pomocí Web Forms (webových formulářů), nebo Windows Forms (formuláře pro Windows). Přes tyto formuláře se přistupuje

k systémovým událostem a službám knihovny CCR. DSS představuje běhové prostředí, které dovoluje službám vyměňovat si zprávy bez ohledu na to, kde v síti se nalézají. Je to tedy prostředí pro komunikaci mezi službami. Na jedné straně jsou služby, které zpracovávají data a provádí výpočty. Na druhé straně jsou služby, které sbírají data ze senzorů a vykonávají příkazy pro ovládání robotických aplikací. Služby jsou funkce, které jsou poskytované jednotlivými knihovnami. Zjednodušená architektura služeb je uvedena na obrázku 2.



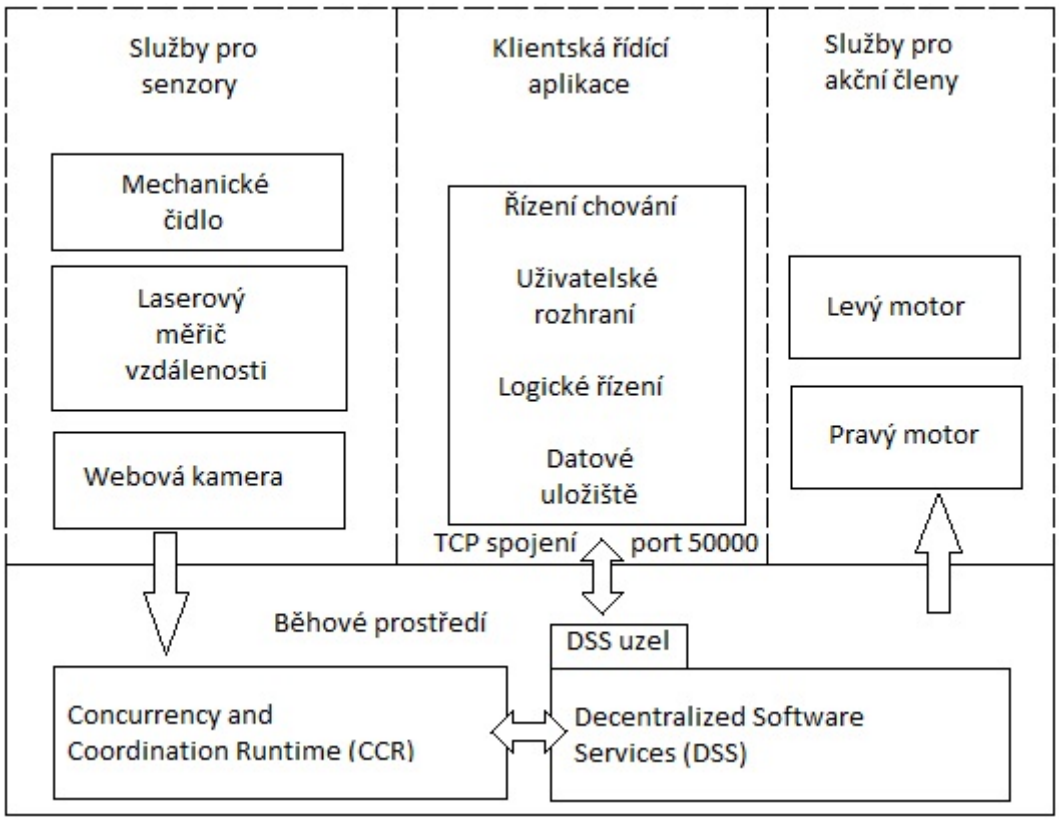
Obrázek 2: Architektura služeb [4]

Každá vytvořená služba má unikátní *identifikátor URI*, který reprezentuje službu v celém prostředí MRDS. Složitější služby mají v hlavičce *reference* na další služby, které jsou ve službě používané. Když vytváříme novou službu nástrojem pro MRDS, jsou v hlavičce importované služby pro čtení manifestů a pro práci s protokolem DSSP. Na *hlavním portu*[1] se připravuje aplikace a získává funkce od služeb. Data jsou prokládána, aby se mohla získávat z více služeb zároveň. Na *porty pro požadavky*, přicházejí požadavky na funkce služeb. V *manipulátoru požadavků* se požadavky třídí a žádají partnerské služby o potřebné informace. Koordinace mezi požadavky na služby a daty odcházejících od služeb probíhá ve stavech služeb.

Schéma komunikace CCR, DSS, služeb a klientské aplikace je na obrázku 3. Služby čtou data ze senzorů a předávají je přes CCR a DSS do logického řízení. V logickém řízení se data zpracovávají a po zpracování dat se posílají řídící příkazy přes DSS a CCR na slu-

[1] Port je fyzické rozhraní, ke kterému se připojuje vnější zařízení. Dělí se na seriový, paralerní, infračervený, gameport a síťový port [10].

žby akčních elementů. Veškeré koordinace služeb řídí běhové prostředí CCR s přispěním DSS, které navíc komunikuje s klientskou řídící aplikací.



Obrázek 3: Architektura běhového prostředí MRDS

2.3 Soubory Manifest

Manifest je soubor založený na otevřeném standardu XML s příponou .xml nebo .Manifest.xml, který obsahuje tagy (značky), které popisují jednotlivé objekty a jejich vlastnosti. Manifesty jsou uloženy především v adresáři `MRDS\samples\Config`. Soubory manifest definují vlastnosti objektů používaných v prostředí MRDS. Manifesty se používají především jako konfigurační soubory a rozdělují se na tři druhy.

2.3.1 Konfigurační Manifest

První druh manifestu se pozná podle párového tagu `<Manifest>`. Tyto manifesty popisují vlastnosti robotického hardware (motor, řídící jednotka, čidlo a další). Především

definují komunikační vlastnosti hardware, jako jsou porty, rychlost přenosu, druh rozhraní, závislosti na služby a na konfigurační soubory. Tento manifest je možné upravovat a vytvářet pomocí editoru DSSME nebo textovým editorem. Tento druh manifestu se vytváří automaticky při exportu projektu z prostředí VPL do projektu MVS.

2.3.2 Manifest pro dokumentaci

Druhý manifest je označen párovým tagem *doc*. Tento manifest popisuje služby definované v knihovnách, které byly vytvořeny při kompilaci nové aplikace. Je to soubor dokumentace, který se používá při generování dokumentačních souborů nástrojem *DssInfo*. Tento manifest se nedá upravovat editorem DSSME, ale lze jej upravovat pomocí textového editoru nebo editoru pro dokumentaci. Soubor se automaticky vytváří při kompilaci aplikace v prostředí MVS nebo při exportu aplikace z prostředí VPL.

2.3.3 Manifest s popisem simulační scény

Třetí druh manifestu je označen párovým tagem *SimulationState*. Tento manifest popisuje stav simulačního prostředí, vložené entity a jejich vlastnosti, chování simulačního prostředí, gravitace a další vlastnosti simulace. Manifest nelze upravovat pomocí DSSME, ale lze jej upravovat pomocí textového editoru nebo načtením do prostředí Visual Simulation Environment (VSE) upravením entit a opětovným uložením *Save Scene As*. Pokud scénu uložíme tímto způsobem, vytvoří se dva manifesty. Jeden obsahuje simulační popis s příponou .xml, který můžeme opět otevřít v prostředí VSE a upravovat. Druhý manifest obsahuje popis referencí na služby. Pomocí něj lze spustit simulaci z příkazové řádky DSSCP. Tento manifest má příponu .Manifest.xml.

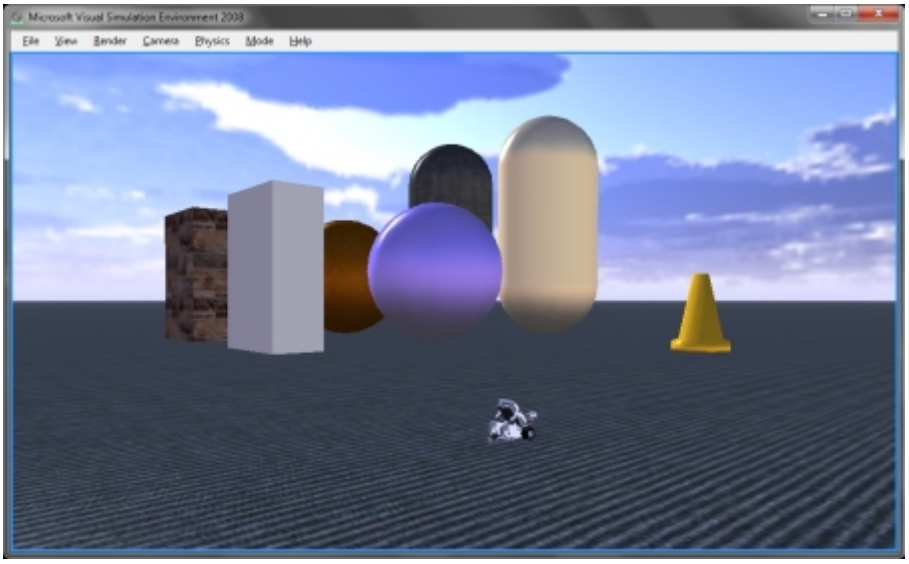
2.4 Visual Simulation Environment

MRDS nabízí pro simulace robotických aplikací simulační prostředí VSE. V tomto prostředí se simuluje reálné chování entit (objektů), se zachováním fyzikálních zákonů. Struktura VSE je založena na *Simulation Engine Service*, která vykresluje entity a výpočet simulačního času. Dále sleduje stav simulačního prostředí a simuluje entity. Další významnou složkou je *Managed Physics Engine Rapper*, která přináší zjednodušené příkazy pro komunikaci s API (rozhraní pro programování aplikace). Jednou z dalších aplikací rozšiřující vlastnosti simulačního prostředí je *AGEIA PhysX Technology*[2]. Tato technologie dokáže používat hardware zvyšující akceleraci a kvalitu simulace chování robotů v simulačním prostředí pomocí *AGEIA PhysX* procesorů. Nejdůležitější částí simulačního prostředí je aplikace *Entites*. Je to aplikace, která v simulačním prostředí reprezentuje roboty a fyzické objekty, jako například stěny, překážky a další simulované předměty. Simulační prostředí je napsané v prostředí XNA, což je prostředí pro programování her

[2] Akcelerátor fyzikálního prostředí [11].

pod Windows.

Simulační prostředí VSE pro MRDS nabízí třídimenzionální (3D) pohled na simulovanou scénu s možností přepínání mezi vloženými kamerami viz obrázek 4. S kamerami je možné pohled plynule přibližovat, oddalovat, plynule měnit úhel pohledu ze všech stran, shora i zdola. Je možné přidávat a editovat entity, při přepnutí simulačního prostředí do editačního módu. Vkládané entity mohou být ve formátu Collada (soubor pro ukládání dat z animačních programů, například z programu SolidWorks). Lze měnit zobrazení z klasické animace na zobrazení modelů s vykreslením hran, nebo zobrazení os x, y, z, každé z entity.

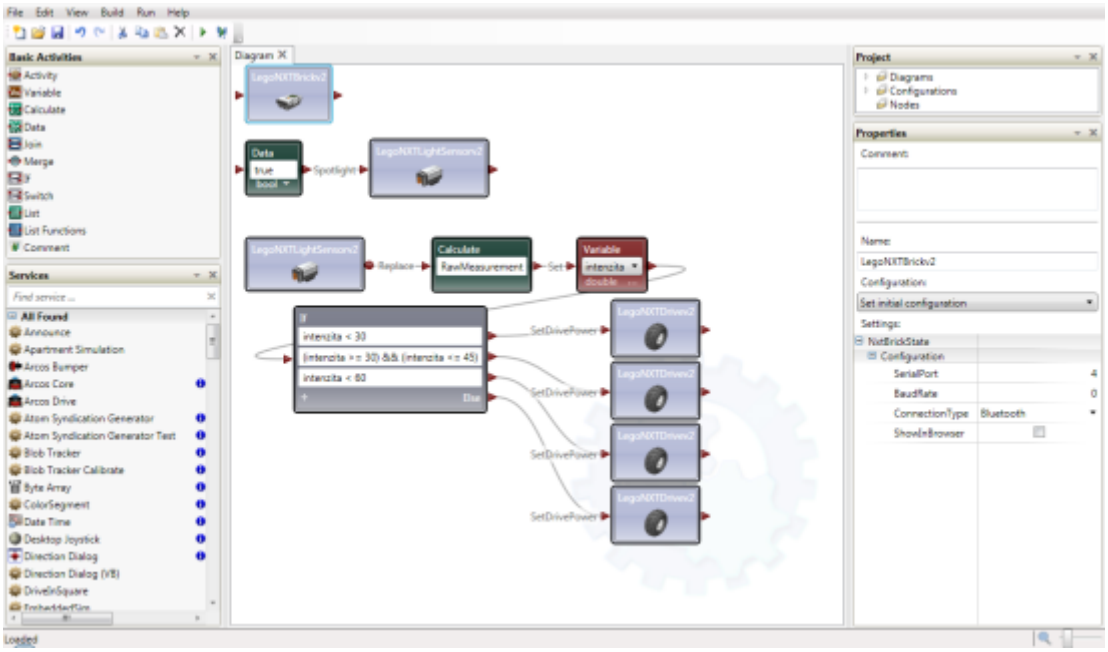


Obrázek 4: Klasické zobrazení animace ve VSE

Vyzkoušet si toto prostředí můžeme ihned po nainstalování MRDS. Zároveň je nainstalováno i několik průvodních simulací pro řadu robotů (NXT, iRobot a další). Text pro tuto kapitolu byl čerpán z [1].

3 Visual Programming Language

Vzhled prostředí VPL je na obrázku 5. Je to vizuální programovací prostředí vyvinuté přímo pro použití služeb z knihoven DSS a CCR. Programování se provádí stylem drag and drop (chyt'a táhni). To znamená, že v programovacím prostředí máme v levém sloupci seznam dostupných služeb a ty vkládáme přetažením na pracovní plochu. Tyto služby lze mezi sebou spojovat a vytvářet tak inteligentní aplikace. V tomto prostředí je možné naprogramovat celou řídící aplikaci. Je určené pro vývojáře, kteří neovládají žádný programovací jazyk, ale používají jej i zkušení programátoři pro rychlé sestavení jednodušších služeb, jejich propojení a export do projektu MVS. Projekt je exportován v jazyce C# a do složky projektu se přidají i všechny potřebné knihovny a manifesty. Je to velmi jednoduchá a rychlá cesta, jak vytvořit nový projekt. Touto cestou si vytvoříme vlastní manifest a projekt se službami pro danou robotickou aplikaci. Pomocí souborů manifest připojíme ke službám součásti robota nebo jejich simulace v prostředí VSE. Problém může nastat, když chceme v prostředí VPL programovat složitější struktury. Především se ve velkém množství služeb můžeme snadno ztratit. Text byl čerpán z [1].



Obrázek 5: Ukázka prostředí VPL

Na obrázku 5 je zobrazeno prostředí VPL s první verzí programu Line Follower. Jak je z obrázku patrné, prostředí je rozděleno do tří sloupců. V levém, jak již bylo zmíněno, se nachází jednotlivé služby a také základní aktivity (if, switch, list, join a další) a pole s chybovými výpisy (upozorňují na chyby vzniklé během programování). V prostředním sloupci jsou jednotlivé záložky s diagramy respektive konfiguracemi. Při rozkliknutí ak-

tivity je zobrazena nová záložka s jejím diagramem (obsahem). Při rozkliknutí služby je zobrazena záložka s její konfigurací. Ve třetím (pravém) sloupci se nachází buňky projekt a vlastnosti. V buňce projekt je seznam vytvořených diagramů[3], konfigurací pro jednotlivé služby a seznam poznámek k projektu. V buňce vlastnosti se zobrazují vlastnosti pro aktuálně označenou aktivitu, službu nebo spojnici mezi jednotlivými aktivitami respektive službami. Buňky v levém a pravém sloupci se dají zcela skrýt a tím zvětšit prostor pro diagram. Tato možnost se hodí například u rozsáhlejších programů, pro jejich úplné respektive částečné, větší zobrazení.

V diagramu na obrázku 5 jsou použity čtyři základní aktivity (Data, Calculate, Variable a If) a tři služby (LegoNXTBrickv2, LegoNXTLightSensorv2 a LegoNXTDrivev2). Službu *LegoNXTBrickv2* je nutné použít v každem program vytvářeném pro NXT robota. Jedná se o hlavní řídící službu, která komunikuje s NXT kostkou. Na tuto službu se pak odvolávají (v nastavení) všechny další použité služby[4]. pro NXT v programu (v tomto případě služba pro světelný senzor a servomotor). Spojnice mezi jednotlivými entitami v diagramu slouží k přenesení nebo nastavení hodnot. Text vyskytující se na konci spojnice (u vstupu aktivity respektive služby) označuje funkci, která je pro toto spojení využita. Do diagramu je můžné vkládat i poznámky a mít tak okomentované důležité části programu a jejich řešení.

3.1 První program

Klasickým a nejpoužívanějším prvním programem, který si většina programátorů vytvoří v pro ně novém prostředí nebo programovacím jazyce, je program *hello world*. I v této práci bude vytvořen jako první program právě zmiňovaný *hello world*. Jedná se o program, který uživateli zobrazí nápis "Hello world!". V tomto případě byl program záměrně modifikován do podoby, kdy je zadaný text vyřčen. Tím si zároveň ověříme, jak zvládá VPL převést a následně zpracovat text do podoby mluveného slova. VPL podporuje pouze znaky anglické abecedy.

Obrázek 6: Ukázka programu Hello world

Na obrázku 6 je program Hello world v prostředí VPL. Jsou zde celkem tři spojené bloky, které byly použity (přetaženy) z levého sloupce v prostředí VPL. Po spuštění

[3]Záložka Diagram je zde vždy, další diagramy pak přísluší vytvořeným aktivitám.

[4]Jednotlivé služby můžeme použít v grafu vícekrát a při každém novém vkládání se nás VPL zeptá, zda chceme přidávanou službu přidat nezávisle na již existující stejné službě nebo jí chceme s ní propojit. V úlohách pro tuto práci byly služby vždy propojeny.

prostředí VPL je vytvořen nový projekt s prázdným diagramem a můžeme tak začít programovat. Budeme potřebovat dvě základní aktivity (dva bloky) a jednu službu. První blok (Data), do kterého je zadán text (může být zadáván i v uvozovkách). Výstup tohoto bloku je pak vstupem druhého, který slouží jako proměnná (Variable). Pomocí propojení těchto dvou bloků jsme vytvořili a naplnili proměnnou (v tomto případě proměnnou *text*). Aby bylo možné tuto proměnnou vypsat respektive slyšet její obsah, musíme využít služby*Text to speech* (TexttoSpeechTTS), která se stará o interpretaci vstupního textu do podoby mluveného slova.

Při propojování jednotlivých bloků, aktivit nebo služeb je vždy vyvoláno nové okno pro upřesnění komunikace. V případě, kdy jsme propojovali první dva bloky (přiřazení obsahu proměnné) bylo nutné ve vyvolaném okně vybrat funkci *SetValue*. Tímto jsme zajistili, že dané proměnné bude nastaven obsah předchozího bloku. Použití funkce*GetValue* je pak pro získání obsahu proměnné. Propojení výstupu druhého bloku se vstupem služby bylo nastaveno jako funkce*SayText* a její vstupní proměnná je *text*. Na výběr je většinou řada funkcí, záleží na programátorovi, jak chce daný blok, aktivitu nebo službu využít.

3.2 Tvorba nové aktivity

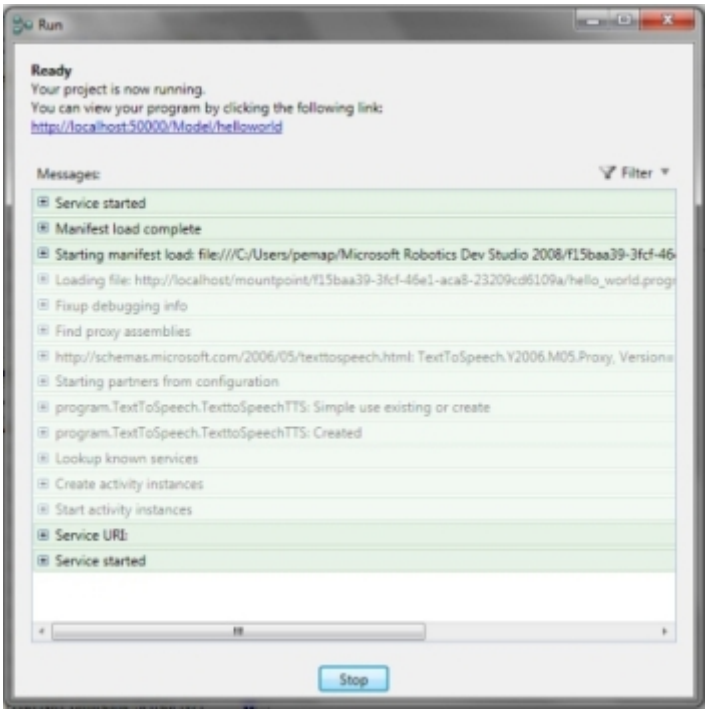
Tvorba nové aktivity probíhá obdobně, jako když přidáváme a propojujeme mezi sebou jednotlivé bloky, aktivity nebo služby v hlavním diagramu. Ve VPL v levém sloupci s částí základních aktivit vybereme blok *Activity* a přetáhneme na hlavní diagram. Kliknutím na aktivitu se v pravé části zobrazí její vlastnosti, zde je možné ji přejmenovat, napsat k ní komentář nebo například její popis. Nás ale zajímá její obsah, na který se dostaneme dvojitým kliknutím. Zobrazí se nový prázdný diagram pojmenovaný podle aktivity. Nyní se můžeme přepínat mezi hlavním diagramem a diagramem pro aktivitu pomocí záložek.

Tvorba obsahu aktivity respektive její naprogramování probíhá obdobně jako programování na hlavním diagramu. Bloky aktivit slouží ve VPL obdobně jako metody respektive funkce v programovacích jazycích jako například C#. Programátor si tak může přehledně rozdělit program do několika navzájem propojených aktivit a na hlavním diagramu pak vidí například jen dvě služby a tři aktivity. Princip využití aktivit je vidět v kapitole 5 u programu *Text on NXT*.

3.3 Spuštění programu

Pokud chceme spustit vytvořený program, můžeme tak učinit dvěma způsoby. Příkaz *Start* (spuštění programu) je možné vyvolat pomocí klávesy F5, nebo přes hlavní menu, kliknutím na záložku *Run* a následným kliknutím na položku *Start*. Po spuštění se objeví nové okno (Run okno), které informuje o právě probíhajících akcích. Jako první se spouští DSS runtime prostředí, dále se pak načítají potřebné manifesty, spouští služby

použité v programu a vytváří lokální schéma, které je možné prohlížet pomocí internetového prohlížeče. Když vše proběhne v pořádku změní se prvotní informace o spouštění DSS runtime prostředí na výpis, že váš projekt byl úspěšně spuštěn a že si jej můžete prohlédnout kliknutím na následující odkaz (odkaz na vytvořené lokální schéma). Veškeré informace vypsané v run oknu se dají jednoduše rozkliknout a zobrazit tak více o konkrétní činnosti. Některé zmíněné výpisy a informace jsou vidět na obrázku 7, který představuje, jak vypadá run okno, spuštěného programu Hello world.



Obrázek 7: Ukázka run okna pro program Hello world

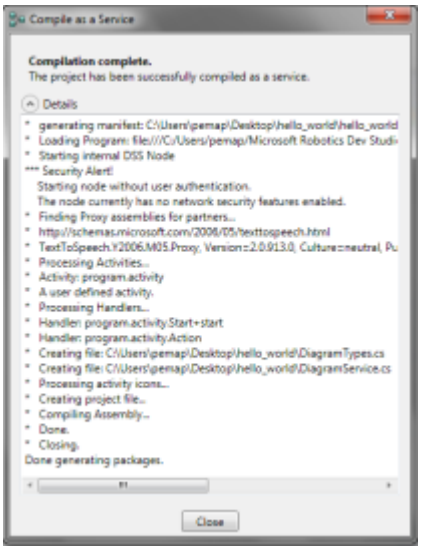
Pokud si chceme zobrazit schéma programu, popřípadě si projít nastavení celého prostředí a jeho služeb, klikneme na vytvořený odkaz. Jako první co po nás bude prohlížeč chtít je autentizace (přihlášení) na stránky. Zde musíme zadat uživatelské jméno a heslo, jaké používáme pro přístup k našemu windowsovskému uživatelskému účtu. Obě kolonky musí být vyplněny, nelze tedy se přihlásit s prázdným heslem (případ, kdy se nepřihlašujeme k windows účtu s pomocí hesla).

Tak jako i v jiných programových prostředích i zde existuje možnost spustit program v debug módu. Funkci *Debug start* je možné vyvolat opět dvěma způsoby, pomocí klávesové zkratky F10 nebo přes hlavní menu, kliknutím na záložku *Run* a následným kliknutím na položku *Debug start*. Proběhne spuštění programu, tak jako u možnosti *Start*, ale v momentě kdy bude celý proces dokoneč (změní se výpis v run oknu), začne se spouštět internetový prohlížeč a bude vyžadovat opět autentizaci. Po přihlášení se zobrazí stránka

s výpisem proběhlé akce a se třemi odkazy. Pod první odkazem *Control Panel* se skrývá seznam všech dostupných služeb v MRDS. Zde si můžeme každou podrobněji prohlédnout respektive dočíst se, jak se jmenuje, k čemu slouží, přiřadit k ní manifest popřípadě jej odebrat. Jedná se vlastně o hlavní ovládací panel MRDS prostředí. Odkaz *Service Directory* poskytuje adresářové služby instancí, které v současné době běží pro spuštěný program. Posledním, třetím odkazem je pak *Debug and Trace Messages* obsahující výpis provedených akcí. Zkrácený výpis z tohoto odkazu je vidět v run okně, ale pro ladění a nalezení chyby, programátorům lépe poslouží detailnější výpis, který se nachází právě pod posledním odkazem. Dají se zde použít automaticky vygenerované filtry a zredukovat tak výpis.

3.4 Diagram jako služba

Jakýkoliv program vytvořený v prostředí VPL lze jednoduše přenést (zkompilovat) do programovacího jazyka C#. Můžeme tak učinit opět dvěma způsoby a to buď klávesovou zkratkou *Ctrl+Shift+B* nebo přes hlavní menu, kliknutím na záložku *Build* a následným kliknutím na položku *Compile as a Service*. Než proběhne zkompilování VPL kódu do C# kódu (ukázka kódu je ve výpisu 2), zeptá se nás program, kam chceme překompilovaný projekt uložit. V compile okně (obrázek 8) je pak obsažen výpis probíhající kompilace (vygenerování manifestu pro aktuální program, načtení programu, vytvoření C# souborů a další). Celý projekt je překompilován z vizuálního jazyka na programovací jazyk a spolu s tím jsou přeneseny i potřebné manifesty použitých služeb. VPL generuje hlavní manifest, pomocí kterého lze pak spouštět vygenerovanou službu. Manifest je pojmenován po kompilovaném projektu (pro program Hello world by se manifest jmenoval hello_world.manifest a jeho náhled je vidět ve výpisu 1).



Obrázek 8: Ukázka compile okna pro program Hello world

```
<?xml version="1.0"?>
<!--This file was created with the Microsoft Visual Programming Language.-->
<Manifest xmlns:texttospeech="http://schemas.microsoft.com/2006/05/texttospeech.html" xmlns:
    diagram="http://schemas.tempuri.org/2010/05/hello_world/diagram.html" xmlns:this="urn:uuid
    :227a78bc-25d9-473b-bab9-b2f1717f9772" xmlns:dssp="http://schemas.microsoft.com/xw
    /2004/10/dssp.html" xmlns="http://schemas.microsoft.com/xw/2004/10/manifest.html">
  <CreateServiceList>
    <ServiceRecordType>
      <dssp:Contract>http://schemas.tempuri.org/2010/05/hello_world/diagram.html</dssp:
          Contract>
      <dssp:PartnerList>
        <dssp:Partner>
          <dssp:Contract>http://schemas.microsoft.com/2006/05/texttospeech.html</dssp:
              Contract>
          <dssp:PartnerList />
          <dssp:Name>diagram:TexttoSpeechTTS</dssp:Name>
          <dssp:ServiceName>this:TexttoSpeechTTS</dssp:ServiceName>
        </dssp:Partner>
      </dssp:PartnerList>
      <Name>this:Diagram</Name>
    </ServiceRecordType>
    <ServiceRecordType>
      <dssp:Contract>http://schemas.microsoft.com/2006/05/texttospeech.html</dssp:Contract>
      <dssp:PartnerList />
      <Name>this:TexttoSpeechTTS</Name>
    </ServiceRecordType>
  </CreateServiceList>
</Manifest>
```

Výpis 1: Ukázka manifestu pro program Hello world

```
namespace Robotics.Hello_world.Diagram
{
    [DisplayName("Hello_world")]
    [Description("A user defined activity.")]
    [dssa.Contract(Contract.Identifier)]
    public class DiagramService : dssm.DsspServiceBase
    {
        // Service state
        [dssa.InitialStatePartner(Optional = true)]
        private DiagramState _state;

        // Service operations port
        [dssa.ServicePort("/Hello_world", AllowMultipleInstances = true)]
        private DiagramOperations _mainPort = new DiagramOperations();
```

Výpis 2: Ukázka C# kódu pro program Hello world

4 LEGO MINDSTORMS NXT 2.0

Zatímco LEGO není třeba nikomu představovat, MINDSTORMS NXT si to zaslouží. V jedné krabici dostanete stavebnici chytrých robotů, které si musíte nejen sestavit, ale také naprogramovat. Z počáteční hromady Lego kostek vyrobíte průzkumné vozidlo, tancujícího robota, útočícího štíra, zkrátka cokoli, na co si jen vzpomenete. Aby to nebyla jen mrtvá hromada kostek, využíváte řídící jednotku obsluhující tři servomotory a čtyři čidla – vzdálenost, světlo, dotyk a zvuk. Kostku naprogramujete na počítači i bez velkých znalostí programovacích jazyků, přičemž ti odvážnější, to mohou zkusit ve Visual Studiu s doplňkem Microsoft Robotics Studio.

Hlavní kouzlo MINDSTORMS NXT je v kreativním přístupu od samého počátku. Nemáte nějakého předchystaného robota, kterého jen oživujete. Sami si musíte vymyslet, jak daný cíl co nejlépe dosáhnout kombinací vhodně pospojovaných dílů a vhodného programování. Neladíte tak jen program, ale upravujete i kostky a rozmístění čidel. Celé je to přitom velmi názorné a jednoduché. Takže se můžete soustředit skutečně na samotný problém, ne na zbytečné překážky.

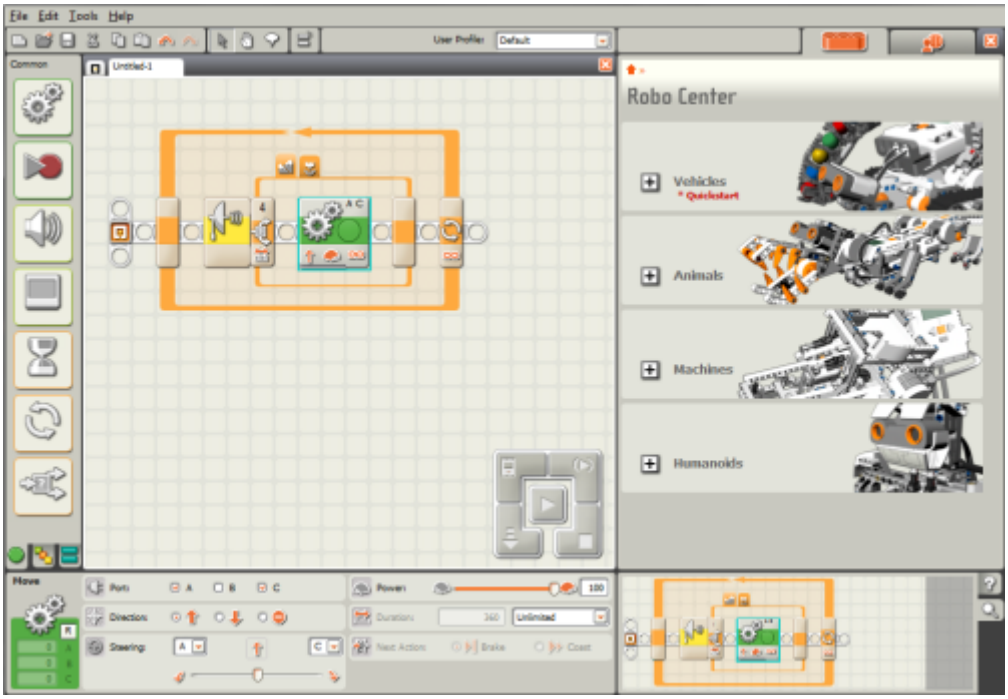
S MINDSTORMS dostanete návody na stavbu a oživení čtyř základních modelů – auta, štíra, robotické ruky a humanoida. Liší se složitostí stavby i programování, po jejich sestavení rychle pochopíte základy a budete je různě vylepšovat. Po pár dnech už budete sami přicházet s novými typy robotů.

NXT není prvním krokem Lega do světa robotiky. Předchozí MINDSTORMS ale měl řadu problémů, motory nebyly synchronizované a udržení robota v přímém směru bylo obtížné, čidla nebyla tak kvalitní a modely dostatečně odolné. Nový NXT využívá robustní pevně propojené TECHNICS díly.

Díky Bluetooth mohou mezi sebou spolupracovat až čtyři řídící jednotky, případně lze program řídit z počítače. Z několika sad stavebnic tak vytvoříte i velmi náročné roboty s mnoha čidly a motory. Text byl převzat z [6].

4.1 NXT-G

Součástí příslušenství k LEGO MINDSTORMS NXT 2.0 je také programovací prostředí NXT-G, pomocí kterého lze snadno, během několika okamžiků, vytvořit jednoduché aplikace a následně je vyzkoušet na robotovi.



Obrázek 9: Ukázka prostředí NXT-G

Na obrázku 9 je ukázka, jak vypadá prostředí NXT-G. Je přehledně rozděleno na menu nabídku a čtyři nepravidelně velké části. První, levá horní část slouží jako diagram. Zde se dají spojovat jednotlivé služby a vytvářet tak z nich plnohodnotné programy pro robota. Jednotlivé služby, jako jsou služba pro servomotor, služby pro jednotlivé senzory nebo například blok pro časovač, smyčku a další, jsou umístěny ve sloupci na levé straně. Stačí si vybrat příslušnou službu a pomocí myši ji přetáhnout na potřebné místo v diagramu.

Každý program začíná od hlavní kostičky v diagramu (kostička s bílým logem MINDSTORMS na oranžovém pozadí), ke které je možno připojovat jednotlivé bloky, služby nebo aktivity ve třech směrech (shora, zprava a zespod). Na zmíněném obrázku je program sestaven na pravou část hlavní kostičky. Je zde použit blok pro nekonečnou smyčku[5], do kterého je umístěn celý zbytek programu. Služba pro ultrazvukový senzor je zároveň rozhodovacím prvkem a podle vzdálenosti objektu před tímto senzorem je pak vyhodnocena podmínka. Poslední službou, která se nachází v diagramu je služba pro ovládání servomotoru (v tomto případě ovládá zároveň dva servomotory a nastavuje jim jízdu vpřed).

Dole v první části se nachází tlačítka pro komunikaci s robotem, první slouží k vyhledání a spojení s robotem pomocí USB respektive Bluetooth. Napravo od tohoto tlačítka je pak tlačítko k nahrání a spuštění právě načteného programu. Stejnou funkci plní i pro-

[5]V prog. jazyce podmínka while(true). V prostředí VPL je tato smyčka implementována automaticky.

střední tlačítko. Spodní levé tlačítko slouží pouze k nahrání programu do paměti robota a poslední je stop tlačítko, využívané k zastavení spuštěného programu v NXT kostce.

Pod částí diagram se nachází část nastavení. Klikne-li uživatel na libovolnou službu, respektive blok, zobrazí se zde jeho příslušné nastavení. Uživatel může jednoduše měnit a přizpůsobovat vlastnosti služeb respektive bloků. Vše je velice přehledně graficky zpracováno, rozděleno a členěno, vytvořit tak a následně nastavit program, nebude problém, ani pro úplné začátečníky.

Vedle části nastavení se nachází informační a náhledová část, členěny do záložek. Na záložce info se nachází odkaz na dokumentaci, která se nainstaluje zároveň s programem. Druhá záložka pak zobrazuje náhled celého diagramu. Pomocí myši a levého tlačítka se může uživatel rychle a přehledně pohybovat po diagramu. Tato funkce se hodí především u delších respektive rozsáhlejších programů, klasické scrollování, ani posun myši do stran po diagramu zde nefunguje.

Zbývá poslední pravá horní část, která je opět, jako předchozí, rozdělena na dvě záložky. Ikonka na první záložce znamená, že se zde nachází zpravidla čtyři demonstrativní programy. Ty jsou kompletně okomentovány a krok po kroku vyobrazeny, jak poskládat, naprogramovat a zprovoznit je. Ve verzi LEGO MINDSTORMS NXT 2.0 jsou na výběr čtyři formy: Vehicles, Animals, Machines a Humanoids. Druhá záložka s názvem "My Portal" obsahuje užitečné odkazy na internetové stránky http://mindstorms.lego.com/.

4.2 NXT kostka

Řídící jednotka, neboli NXT kostka, popřípadě také mozek robota, jsou různé alternativy, jako pojmenovat tuto chytrou krabičku, starající se o oživení celého robota a vykonávání tak nejrůznějších činností. Můžeme zde mít uloženo několik menších programů respektive jeden větší, rozsáhlejší a to z důvodu malé kapacity paměti. Tato nepříjemná vlastnost se projevuje zejména při komunikaci kostky s počítačem, kdy se postupně zaplňuje fronta zpráv a při jejím naplnění se automaticky začnou ztrácet nejstarší zprávy. NXT kostka podporuje pouze znaky anglické abecedy.

Na spodním okraji, pod nápisem NXT, se nachází čtyři očíslované vstupní porty. Zde se připojují veškeré senzory, vyvinuté pro NXT. Libovolný senzor se dá připojit do libovolného portu, ovšem uživatel na tuto skutečnost musí pamatovat, používá-li demonstrační programy, které zachovávají klasické zapojení (senzory respektive servomotory a jejich čísla respektive znaky portů, jsou popsány v následující podkapitole). Chceme-li připojit servomotory, potřebujeme tedy výstupní porty, najdeme je v horní části NXT označeny symboly A, respektive B, respektive C.

Pro komunikaci s počítačem, je zde použito rozhraní USB respektive Bluetooth. Obě tyto rozhraní, má-li některé z nich spojení například s počítačem, jsou zobrazena pomocí

ikonek v levé horní části displaye, které se mění v závislosti na aktivním spojení s druhým zařízením. Zařízení Bluetooth si je schopno zapamatovat až čtyři spojení, jak již bylo výše zmíněno, mohou spolu tedy komunikovat až čtyři tyto řídící jednotky. Rozhraní USB se nachází v horní části kostky, hned vedle výstupních portů.

Uprostřed displaye v jeho horní části je zobrazeno aktuální pojmenování řídící jednotky. Tento název je možné kdykoliv změnit, jak v prostředí NXT-G, tak ve VPL. Napravo od pojmenování řídící jednotky se nachází ikonka práce NXT, po které obíhá prázdný pixel v případě, že NXT kostka pracuje správně. V opačném případě je nutno ji restartovat. Hned vedle indikace stavu je vyobrazena ikona pro stav baterie. Pod ikony se nachází čára, která odděluje horní část displaye od prostoru, kde se vykresluje právě spuštěná aplikace nebo se zde prochází menu.

NXT kostka obsahuje pár předpřipravených programů, pomocí kterých lze po sestavení robota ihned rozpohybovat. Je zde možnost vytvářet jednoduší programy i přímo na display. Programy vytvořené v NXT-G zůstávají uloženy v paměti kostky, dokud je sami neodstraníme, můžeme je tak kdykoliv spustit, anižbychom museli mít propojeno NXT s počítačem. Naopak programy vytvářené pomocí VPL jsou ukládány do dočasné paměti a po přerušení spojení s počítačem, zamrznou na právě vykonávané akci a většinou je nutno NXT restartovat a program se odstraní.

Pod displayem se nachází čtveřice tlačítek. Levá a pravá šipka, které slouží k procházení menu NXT. Prostřední tlačítko, žluté, sloužící jako enter. Tímto tlačítkem se NXT zapíná nebo potvrzují události. Čtvrté a poslední tlačítko funguje jako storno, tlačítko zpět, nebo k vyvolání akce na vypnutí NXT. Nalevo od tlačítek a displaye je, na boční straně pod plastovým povrchem, řídící jednotky umístěn malý reproduktor.

Obrázek 10: NXT kostka [9]

Technická specifikace pro NXT kostku [5]:

- 32 bitový mikroprocesor ARM7
- 256 KB FLASH, 64 KB RAM
- 8 - bitový mikroprocesor
- 4KB FLASH, 512 B RAM
- Bluetooth® komunikace, Bluetooth třídy II V2.0 compliant
- USB 2.0 port
- 4 vstupní porty, 6 vodičová digitální platforma
- 3 výstupní porty, 6 vodičová digitální platforma
- maticový display, 60 x 100 pixelů
- reproduktor, 8 KHz kvalita zvuku
- elektrické zdroje: nabíjecí lithiová baterie popřípadě 6 AA článků
- připojení na síť: USA: 120 VAC, 60 Hz ; UK, EU, AUS: 230 V, 50 Hz

4.3 Senzory

V následujících podkapitolách budou popsány jednotlivé senzory, které byly použity při testování vytvořených aplikací. Senzory vyrábí dvě společnosti, jednou z nich je právě společnost Lego a druhou, která má její podporu je pak HiTechnic. Pro tuto práci však byly použity senzory od firmy Lego, které jsou součástí základního balení stavebnice LEGO MINDSTORMS NXT 2.0.

4.3.1 Dotykový senzor

Tento senzor umožňuje robotovi reagovat na předměty nacházející se v nejbližším okolí před ním. Senzor má dvě polohy, funguje obdobně jako vypínač, kdy stlačení většinou znamená, že je v poloze zapnuto a naopak, ve standardním, nestlačeném stavu, je v poloze vypnuto (ovšem záleží na uživateli, jak nastaví chování tohoto senzoru).

Využití se nabízí například u verze robota humanoida, kdy pomocí dvou stejných senzorů je ovládána koordinace nohou. V aplikacích vytvořených k této práci, jsou využity dva tyto senzory pro ovládání vozítka, jak bude popsáno v následující 5. kapitole.

- **Označení:** Touch Senzor
- **Označení VPL:** Lego NXT Touch Sensor (v2)
- **Port pro připojení k NXT:** 1

Obrázek 11: Dotykový senzor [7]

4.3.2 Zvukový senzor

Zvukový senzor detekuje intenzitu zvuku v decibelech (dB) od jemných a tichých zvuků, až po zvuky hlasité. Zvukový senzor pracuje jak s dB, tak s dBA:

- dB – všechny snímané zvuky včetně vysokých anebo nízkých frekvencí, které lidské ucho neslyší
- dBA – pouze zvuky, slyšitelné lidským uchem

Zvukový senzor měří akustický tlak do úrovně přibližně 90 dB, což odpovídá hluku běžící sekačky na trávu. Údaje o zvuku, který je načítán, jsou LEGO® MINDSTORMS® NXT zobrazovány v procentech tohoto rozsahu.

Pro příklad uvádíme:

- 4-5% odpovídá tichu v obývacím pokuji,
- 5-10% odpovídá vzdálenému hovoru,
- 10-30% je hovor v blízkosti senzoru,
- 30-100% odpovídá hlučnosti v prostředí s hlasitou hudbou.

Tyto rozsahy odpovídají zvukům ve vzdálenosti 1m od senzoru. Text převzat z [5].

- **Označení:** Sound Sensor
- **Označení VPL:** Lego NXT Sound Sensor (v2)
- **Porty pro připojení k NXT:** 2

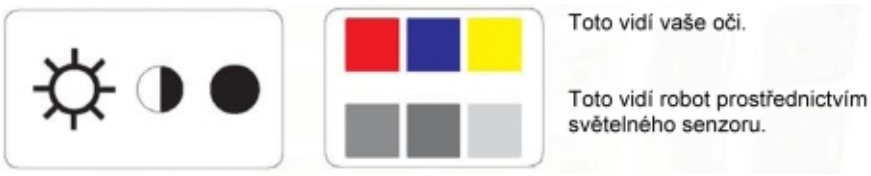
Obrázek 12: Zvukový senzor [7]

4.3.3 Světelný senzor

Světelný senzor je jedním ze dvou senzorů, které robotu umožňují vidění. Světelný senzor umožňuje robotu rozlišit světlo a tmu. Dokáže poznat intenzitu světla v místnosti a změřit intenzitu světla barevných povrchů. Text převzat z [5].

- **Označení:** Light Sensor
- **Označení VPL:** Lego NXT Light Sensor (v2)
- **Port pro připojení k NXT:** 3

Obrázek 13: Světelný senzor [7]



Obrázek 14: Ukázka zpracování barev pomocí light senzoru

4.3.4 Barevný senzor

Druhým senzorem, který umožňuje robotovi vidět, je právě barevný senzor. Ten dokáže rozeznat až šest druhů barev (černá, bílá, červená, modrá, žlutá a zelená). Další dvě

funkce, které tento senzor dokáže, jsou stejné jako u světelného senzoru a to rozpoznání intenzity světla v místnosti a změření intenzity světla barevných povrchů.

Barevný senzor nabízí ještě jedno využití a to v podobě barevné lampičky, kdy je možné na něm rozsvítit červenou, respektive zelenou, respektive modrou barvu (RGB).

- **Označení:** Color Sensor
- **Označení VPL:** není podporován
- **Port pro připojení k NXT:** 3

Obrázek 15: Barevný senzor [8]

4.3.5 Ultrazvukový senzor

Ultrazvukový senzor umožňuje robotu vidět, hledat předměty, vyhýbat se překážkám, měřit vzdálenost a zaznamenávat pohyb.

Ultrazvukový senzor využívá stejných vědeckých principů jako netopýři: měří vzdálenost na základě výpočtu doby, během níž dorazí k předmětu zvuková vlna a znovu se vrátí - stejně jako ozvěna.

Ultrazvukový senzor měří vzdálenost v centimetrech i palcích a zobrazuje ji na displeji. Dokáže změřit vzdálenost od 0 do 255 cm s přesností +/- 3 cm.

Nejlépe se získávají data o předmětech velkých rozměrů. Předměty vyrobené z měkkých materiálů a zaoblených tvarů (např. míče) nebo předměty, které jsou příliš tenké nebo malé, hledá senzor obtížněji. Je-li v místnosti dva a více ultrazvukových senzorů, mohou se vzájemně rušit. Text převzat z [5].

- **Označení:** Ultrasonic Senzor
- **Označení VPL:** Lego NXT Ultrasonic Sensor (v2)
- **Port pro připojení k NXT:** 4

Obrázek 16: Ultrazvukový senzor [7]

4.3.6 Servomotor

Velice důležitou součástí každého robota je minimálně jeden servomotor, který zajišťuje jeho pohyb. Standardně jsou dodávány tři servomotory, kdy dva z nich jsou většinou použity k pohybu robota (jízda respektive chůze). Servomotor lze, kromě níže zmíněného otáčení o určitý počet stupňů, také nechat otáčet určitý časový interval nebo nechat rotovat respektive ujet určitou vzdálenost.

Vestavěný rotační senzor

Každý motor má vestavěný rotační senzor, což umožňuje přesnější ovládání robota. Rotační senzor měří otáčení motoru ve stupních nebo celkové otáčení (s přesností +/- jeden stupeň).

Jedno otočení odpovídá 360 stupňům, takže pokud nastavíte motor na otočení o 180 stupňů, provede jeho hřídel půl otáčky. Text převzat z [5].

- **Označení:** Motor Service
- **Označení VPL:** Existují dvě služby, využívající servomotor: Lego NXT Drive (v2) a Lego NXT Motor (v2)
- **Porty pro připojení k NXT:** A, B, C

Obrázek 17: Servomotor [7]

5 Výčet a specifikace řešených dílčích úloh

V této kapitole budou popsány jednotlivé úlohy, které byly vytvořeny pro tuto práci. Všechny byly následně otestovány na robotovi NXT. Postupně bude představeno deset programů, kde budou využity všechny dříve zmíněné senzory a servomotory. Jelikož VPL je zaměřeno obecně na různé typy robotů, nebylo možné v něm otestovat například barevný senzor (color senzor) nebo možnost víceřádkového výpisu na display. Je to dáno především tím, že předchozí verze LEGO MINDSTORMS NXT tento senzor neobsahovala. Navíc v době vydání MRDS 2008 (libovolné distribuce) nebylo LEGO MINDSTORMS NXT 2.0 ještě vydáno. Proto bylo využito kombinace VPL a NXT-G. Pro srovnání je jedna úloha implementována jak ve VPL, tak v NXT-G.

5.1 Hardware a Software

5.1.1 Použitý software

Pro tuto práci byly použity, dvě vývojová prostředí, jako hlavní na které je práce zaměřena je MRDS verze 2008 Academic Edition a druhé NXT-G. Jelikož je MRDS zaměřeno obecně na roboty, nepodporuje tak veškeré použité senzory nebo jejich vlastnosti. Bylo proto nutné některé programy popřípadě jejich části napsat v NXT-G. Obě tyto prostředí byly nainstalovány pod operačním systémem Windows Vista SP2 a později pod Windows 7. Jako výchozí prohlížeč pro zobrazení stavu programu, senzorů a služeb, byl použit FireFox 3.5+.

5.1.2 Použitý počítač

Pro tuto práci byl použit notebook Acer Aspire 5930G s konfigurací Intel Core 2 Duo 2.0GHz, 3GB RAM, 250GB HDD a NVIDIA GeForce 9600M GT 512MB.

5.1.3 Použitý robot

Pro otestování vytvořených aplikací byl použit reálný robot LEGO MINDSTORMS NXT ve verzi 2.0, jehož jedna z několika možných podob je zobrazena na obrázku 18. Jedná se o stavebnici od firmy Lego s využitím robotických součástí, jako jsou 3 servomotory, dotykový, ultrazvukový, světelný, barevný a zvukový senzor. Více o popisu jednotlivých částí robota a jeho specifikaci jste se mohli dočíst v předchozí 4. kapitole.

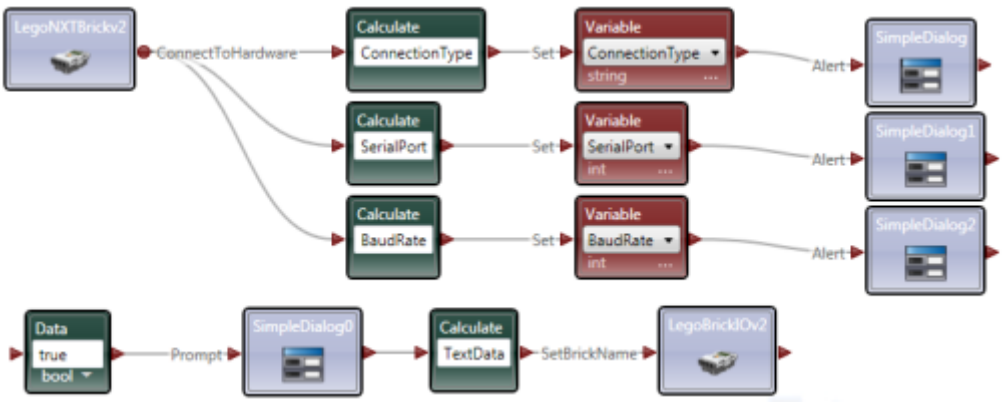
Obrázek 18: Robot LEGO MINDSTORMS NXT 2.0 jako Vehicles [7]

5.2 Info VPL

První úvodní program pro VPL zobrazí pomocí tří dialogových oken informace o rozhraní, přes které robot komunikuje s počítačem (ConnectionType)[6]. Číslo COM portu, na který je robot připojen (SerialPort) a hodnotu změn signálu za jednu sekundu (BaudRate). Tuto hodnotu je možné nechat defaultně nastavenu na nulu.

Při spuštění programu se zároveň zobrazí prompt okno (vstupní potvrzovací okno), do kterého je možno zadat název pro NXT kostku. Po potrzvení se název do ní odešle a okno se zobrazí znovu. Uživatel si tak může měnit pojmenování svého robota znovu a znovu.

Program by se dal rozdělit na dvě části, kdy horní by byla výstupní. Z NXT kostky se zjistí tři zmíněné informace. Vstupní, druhá část pak obstarává možnost přejmenování NXT kostky.



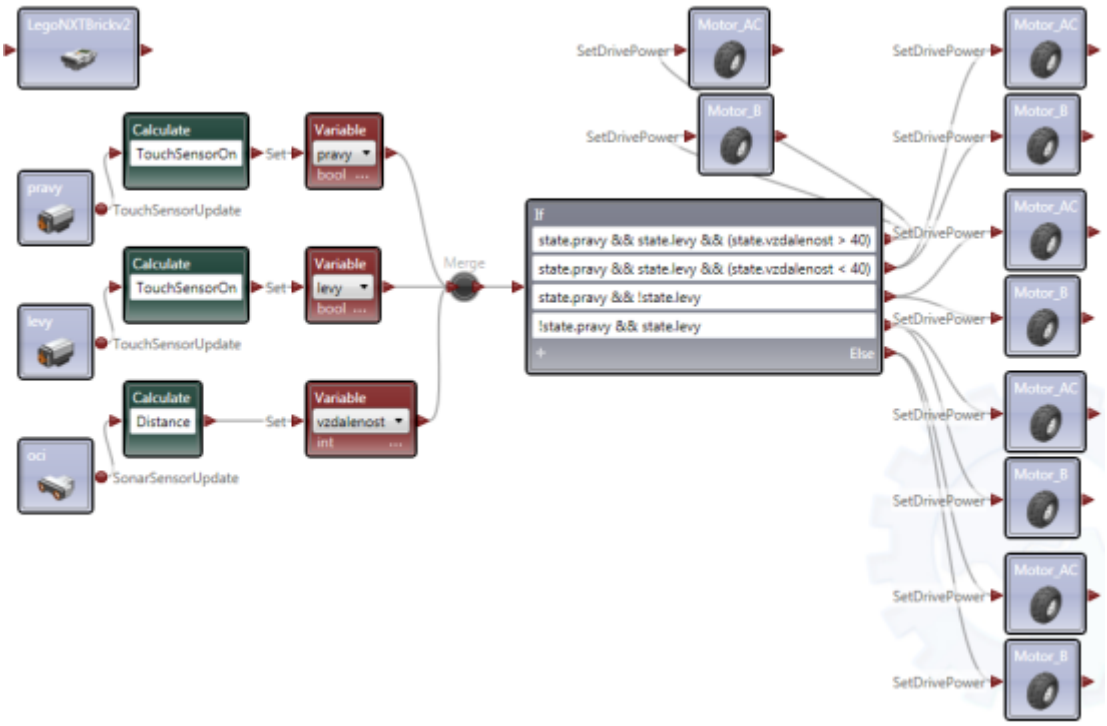
Obrázek 19: Schéma úlohy Info VPL

[6] V prostředí MRDS jedině pomocí Bluetooth, v prostředí NXT-G možno i prostřednictvím USB

5.3 Shooter

Program Shooter (střelec) využívá dvou dotykových senzorů a jednoho ultrazvukového. Dále byly využity všechny tři servomotory. Dva rozpohybují robota, podle kombinace dotykových senzorů a třetí slouží k vystřelování barevných kuliček ze zásobníku. Jak již bylo zmíněno, robot je ovládán pomocí dvou dotykových senzorů. Pokud se oba dva nacházejí v poloze vypnuto (nejsou stisknuty) robot stojí. Při současném stisknutí, nachází-li se tedy oba v poloze zapnuto, se robot rozjede směrem vpřed. Pokud je zmáčknut pouze jeden senzor, robot se točí na stranu, na které se nachází zmáčknutý senzor.

Ultrazvukový senzor přitom po celou dobu, kdy je robot v provozu zjišťuje, jak daleko se nachází předměty (překážky) před ním. Informace o vzdálenosti překážek jsou pak vyhodnocovány v podmínce, jako i stavy dotykových senzorů, viz obrázek 20. Pokud bude před robotem překážka vzdálena méně než 40cm zastaví se a na řadu přichází třetí servomotor. Ten se roztočí na 80% své maximální rychlosti a robot tak postupně vystřeluje jednu kuličku za druhou. K tomu, aby se robot zastavil před překážkou a spustil se třetí servomotor, je zapotřebí mít stisknuty oba dotykové senzory. Stisknutím pouze jednoho senzoru, přestane robot střílet a začne se otáčet do patřičné strany. Obrázek 20 je pak ve větším náhledu v příloze A jako obrázek 31.

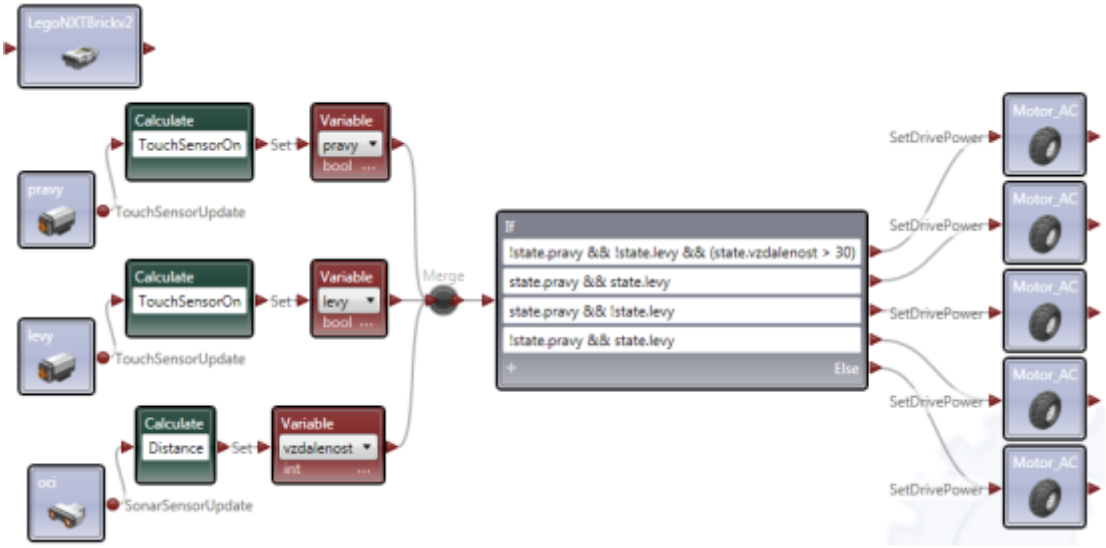


Obrázek 20: Schéma úlohy Shooter (náhled)

5.4 Drive

Pro demonstraci čistě jen pohybujícího se robota byl vytvořen tento program. Po spuštění se NXT automaticky rozjede vpřed a pokud nenarazí na žádnou překážku před sebou, nezastaví se. Pokud ovšem bude před ním předmět ve vzdálenosti menší než 30cm zastaví se. I zde je využito, jako u předchozí úlohy, dvou dotykových senzorů a jednoho ultrazvukového. Při stisknutí obou dvou dotykových senzorů začne robot couvat. Při stisknutí pouze jednoho, dojde k otáčení robota do strany, na které se nachází stisknutý senzor.

Jelikož musí být dotykové senzory připojeny k NXT je nutné se za ním postupně posunovat. Pokud bychom chtěli vylepšit způsob ovládání robota a mít jej na dálkové ovládání, museli bychom použít druhou NXT kostku. Jelikož NXT kostky mohou mezi sebou komunikovat prostřednictvím Bluetooth, mohli bychom tak našeho robota ovládat pomocí tlačítek na druhé NXT kostce.

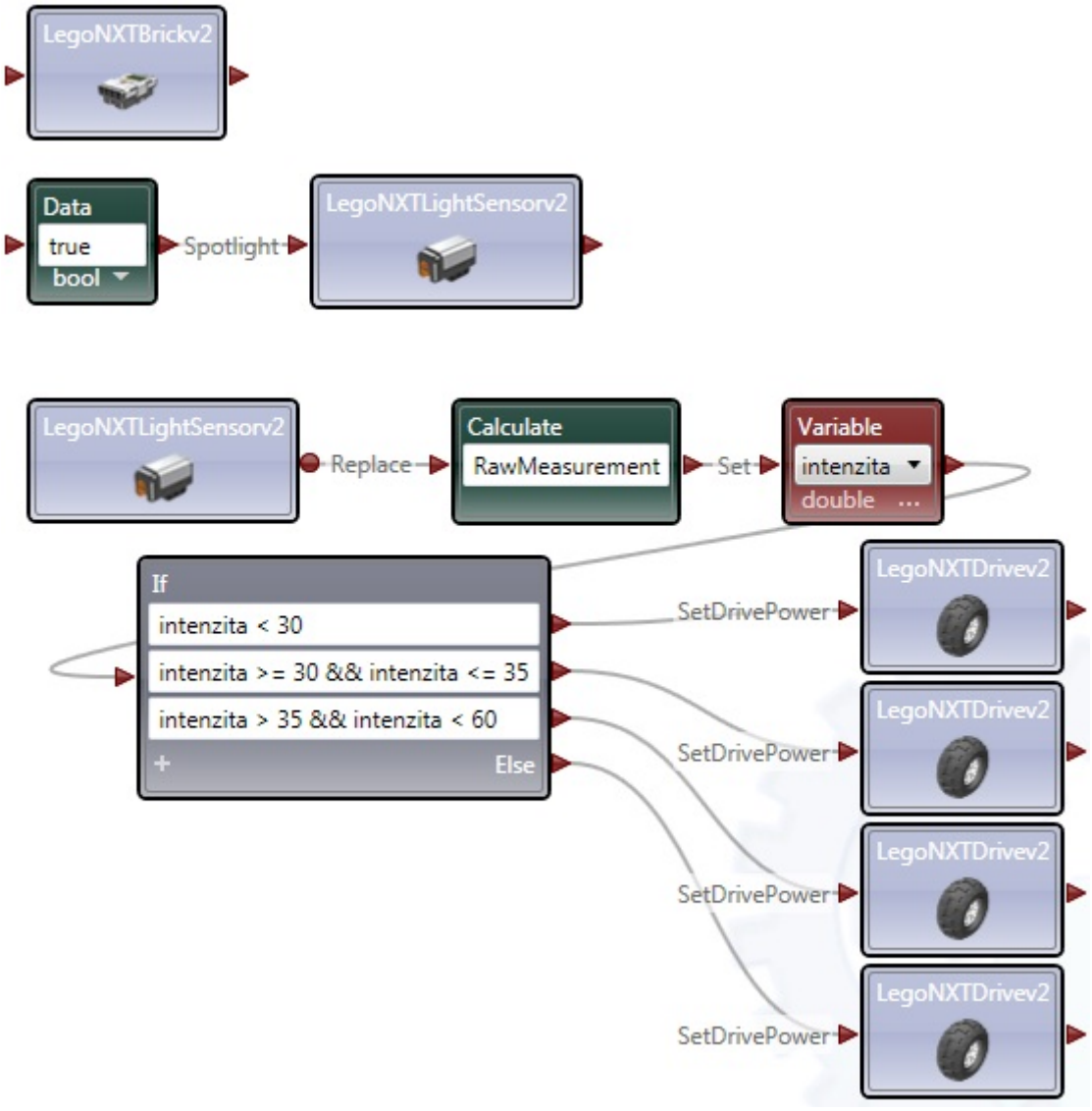


Obrázek 21: Schéma úlohy Drive

5.5 Line Follower

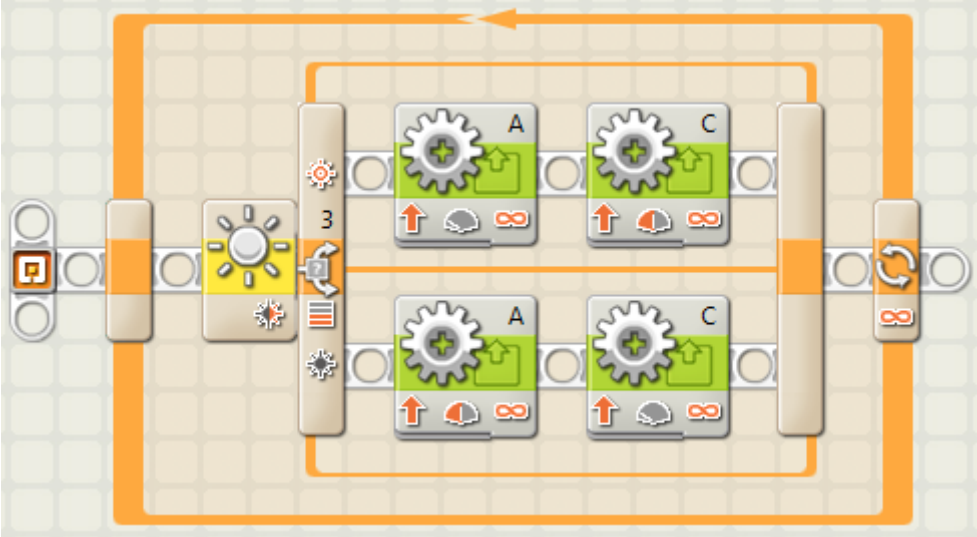
Česky řečeno sledování čáry, je program, pomocí kterého se robot pohybuje podél vyznačené čáry. Současně s robotem je i v balení podložka formátu A2, na které je vyznačeno několik barevných kostek a čar. Mimo jiné je zde vyznačena elipsa, podél které může robot jezdit. Tento program byl napsán jak v prostředí VPL, tak v NXT-G. Budou představeny tedy dvě verze tohoto programu. Rozdíl ve zpracování a obou prostředí je patrný z obrázků 22 a 23. Tento program byl překompilován do C# kódu (viz příloba B).

Robot se pohybuje podél čáry pomocí světelného senzoru. Ten snímá intenzitu jasu barvy a podle podmínek se pak vyhodnotí, který z motorů zrychlí, respektive zpomalí, aby bylo zajištěno postupné sledování čáry. Jde o metodu zig-zag, kdy se robot pohybuje z jedné strany na druhou a zároveň pomalu dopředu a zjišťuje, zda se nachází vedle nebo na čáře. Pro VPL jezdí robot po vnější straně čáry a pro NXT-G naopak po vnitřní straně. Plynulost jízdy robota se v obou případech liší, což je dáno i tím, kdy NXT-G programy jsou uloženy v paměti robota a programy VPL komunikují s robotem v reálném čase pomocí Bluetooth. Zpoždění zpráv je i zde patrné.



Obrázek 22: Schéma úlohy Line Follower pro VPL

Světelný senzor zjišťuje intenzitu jasu (RawMeasurement), která je následně ukládána do proměnné. Na základě hodnot v proměnné dojde k vyhodnocení v rozhodovacím bloku a následně zrychlování respektive zpomalování otáčení servomotorů. Pro zprovoznění (zapnutí) světelného senzoru je nutné do něj poslat booleovskou hodnotu (true)[7].



Obrázek 23: Schéma úlohy Line Follower pro NXT

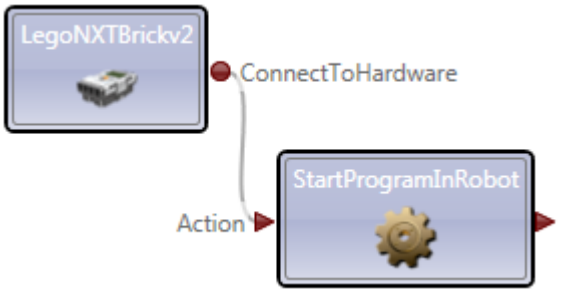
Na obrázku 23 je zobrazeno schéma zapojení v NXT-G. Ve VPL je nutno zadat minimálně tři podmínky pro správnou funkčnost, v NXT-G postačí jeden rozhodovací blok (podmínka). V NXT-G byla nastavena intenzita jasu pro tento případ na 50%.

5.6 Start program

Pomocí tohoto programu můžeme přes VPL spustit libovolný program, který se nachází v paměti robota. Po spuštění aplikace Start program se objeví prompt okno (vstupní potvrzovací okno), do kterého uživatel zadá název programu, který chce v NXT spustit. Název musí mít příponu .rxe, jedná se o koncovku pro NXT programy nahrané v paměti robota. Pokud je zadaný program uložen v paměti NXT spustí se, v opačném případě dojde k zobrazení dialogového okna s chybovým výpisem. Jelikož Start program běží v nekonečné smyčce, prompt okno se znovu zobrazí během několika okamžiků, po potvrzení spuštění vybraného programu. Na obrázku 24 je zobrazena služba LegoNXTBrickv2 propojená s aktivitou StartProgramInRobot.

[7] V prostředí NXT-G, v nastavení pro světelný senzor, stačí zakliknout jeden ze dvou radiobuttonů (On a Off). V prostředí VPL sice tato vlastnost je také, ale pro správnou funkčnost světelného senzoru, je nutné mu ještě navíc poslat booleovskou proměnnou (jinak by senzor vracel stále stejnou hodnotu intenzity barvy).

Aktivita StartProgramInRobot je vidět na obrázku 32[8]. Veškerý její obsah mohl být sestaven již v hlavním diagramu, ale pro přehlednost, rozdělení a ukázku, jak sestavit a využít aktivitu, byla část diagramu přesunuta právě do nové aktivity (StartProgramInRobot). Je zde řešeno nejen spouštění vybraného programu (funkce StartProgram), ale i kontrola, zda již nějaký program v NXT není spuštěn (funkce QueryRunningProgram). Pokud již nějaký běží, proběhne kontrola, zda není spuštěn program, který chceme spustit (podmínka Program == state.ProgramName). Jestliže chceme spustit jiný program (podmínka byla vyhodnocena jako false), je nutné nejdříve zastavit (funkce StopProgram) aktuálně spuštěný program. O této skutečnosti je uživatel informován pomocí dialogového okna. Při úspěšném ukončení respektive spuštění vybraného programu se zobrazí další dialogové okno s výpisem úspěšnosti provedené akce. Pro tento program nebylo nutné využít výstupu aktivity. Pokud bychom jej chtěli mít využit, posílaly by se na něj například informace, zda byl vybraný program spuštěn. Služba pro zobrazení dialogového okna by se pak nacházela v hlavním diagramu propojená s aktivitou StartProgramInRobot. Podmínka Files.Count == 0 zjišťuje, zda je v NXT uložen nějaký program.

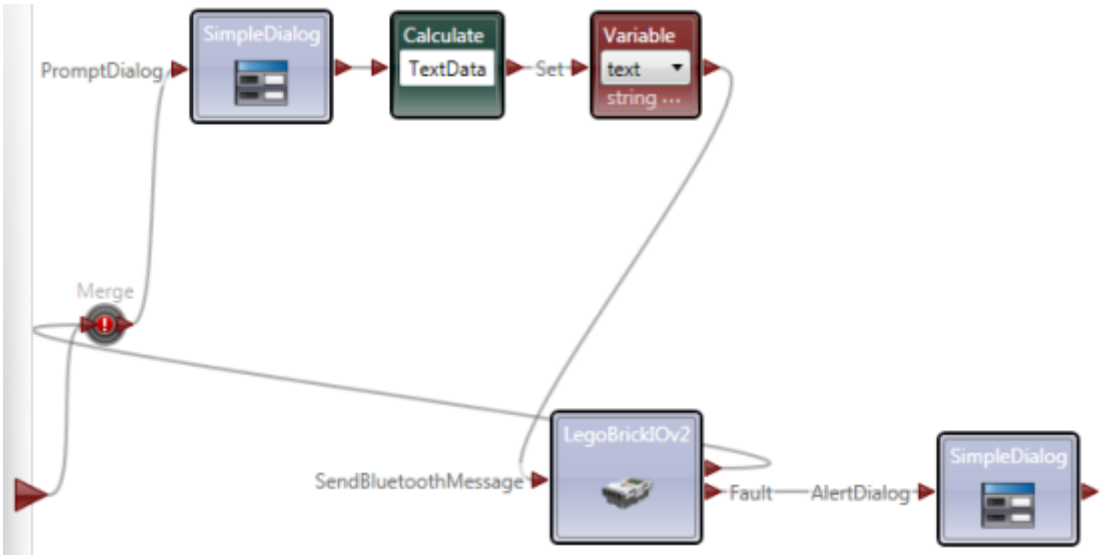


Obrázek 24: Schéma úlohy Start program

5.7 Text on NXT

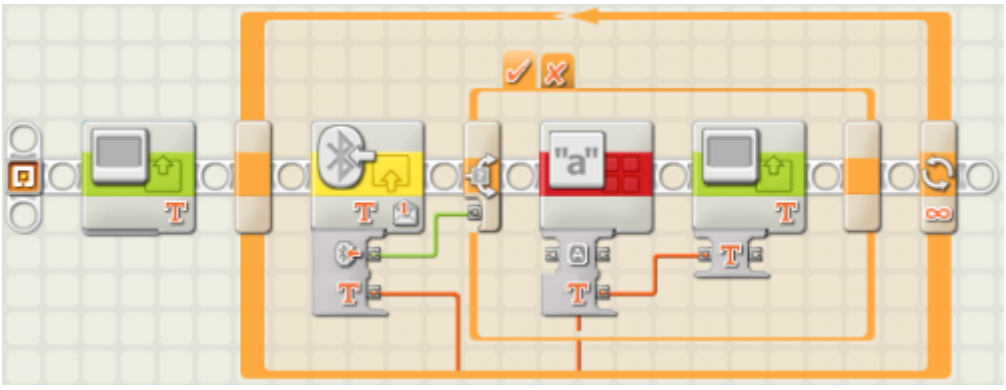
Program pro posílání textu na display NXT vychází z předchozího programu, kdy pomocí Start program šlo spustit libovolný program uložený v paměti robota. Program, který se spouští přes VPL v NXT je zde volán a zadán napevno. Funkce pro zadání názvu programu z předchozí aplikace byla modifikována na funkci zadání textu, který bude následně odeslán pomocí Bluetooth do NXT. Nevýhoda tohoto programu a vlastně zpracování v NXT je, že zadáním nepřiměřeně dlouhého textu dojde k neúplnému zobrazení.

[8]Obrázek je umístěn v příloze A.



Obrázek 25: Schéma aktivity SendText

Na obrázku 25 je zobrazen detail aktivity pro zasílání textu do NXT kostky. Po každém odeslání textu je zobrazeno opět prompt okno (vstupní potvrzovací okno), uživatel tak může stále posílat text. Při selhání odesílání textu se zobrazí chybová hláška. K odeslání textu slouží funkce SendBluetoothMessage, která přijímá jako svůj parametr hodnotu proměnné text. Jak je z obrázku patrné, výstup z prompt okna je získáván pomocí aktivity Calculate, kde je zjišťována hodnota pro TextData. Tato hodnota (obsah prompt okna) je pak ukládána do proměnné text (aktivita Variable).



Obrázek 26: Schéma přijetí úlohy Text on NXT pro NXT-G

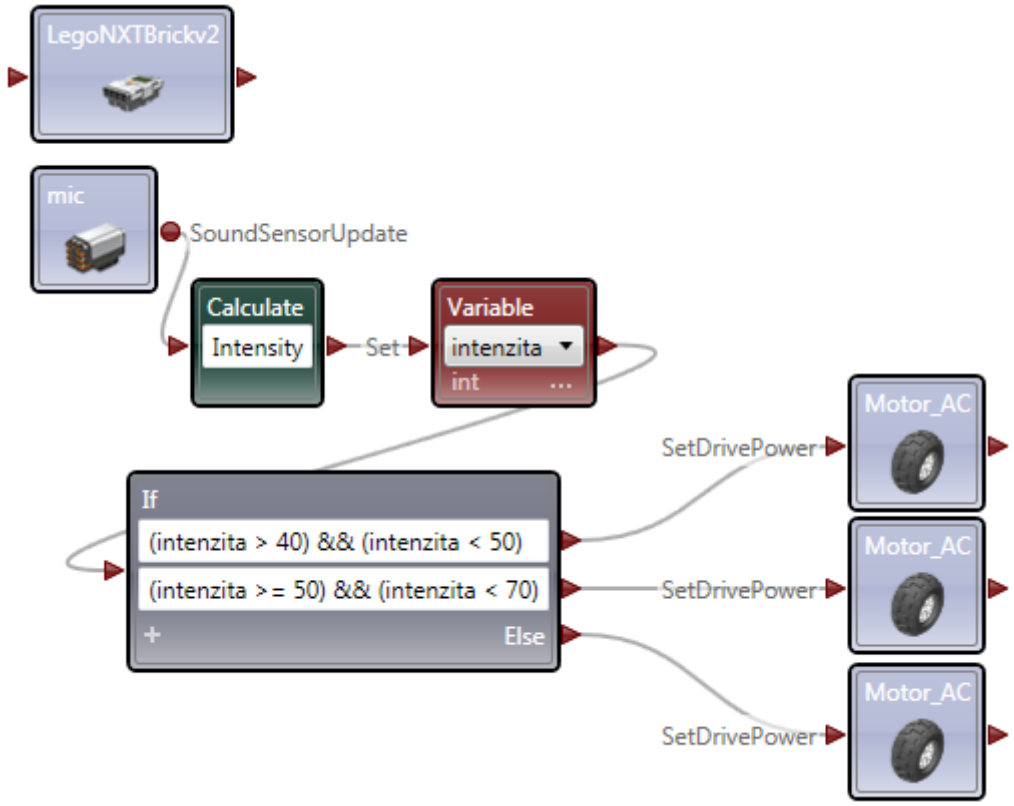
Obrázek 26 zobrazuje schéma zapojení pro příjem a následné zobrazení textu prostřednictvím technologie Bluetooth. Jako první se provede vyčištění displaye, veškerý text na něm zmizí a display zůstane prázdný. Následuje nekonečná smyčka, která kontroluje,

zda není přijímána nějaká zpráva (náš text). Pokud ano, provede se její přijetí, následné zpracování (obsah z mailboxu se pošle do textového bloku) a ten předá již zpracovaný text (string) display, který jej zobrazí.

5.8 Sound

K otestování zvukového senzoru byla naimplementována tato úloha Sound. Jak již bylo zmíněno v předchozí kapitole o NXT (kapitola 4), zvukový senzor snímá zvuk okolí. Z něj pomocí VPL zjistíme, v jak velkém hluku se robot nachází, a podle toho se rozjede nebo zastaví, respektive zůstane stát.

Při testování se robot rozjel až při 40dB. Jsou zde celkem tři podmínky, které rozlišují, jak rychle robot pojede. Pokud bude intenzita hluku v rozmezí 40 až 50db roztočí se servomotory na 80% své maximální rychlosti. Pokud se hluk zvýší, tedy bude minimálně 50dB, ale při tom nepřesáhne hodnotu 70dB, servomotory budou pracovat na 100%[9]



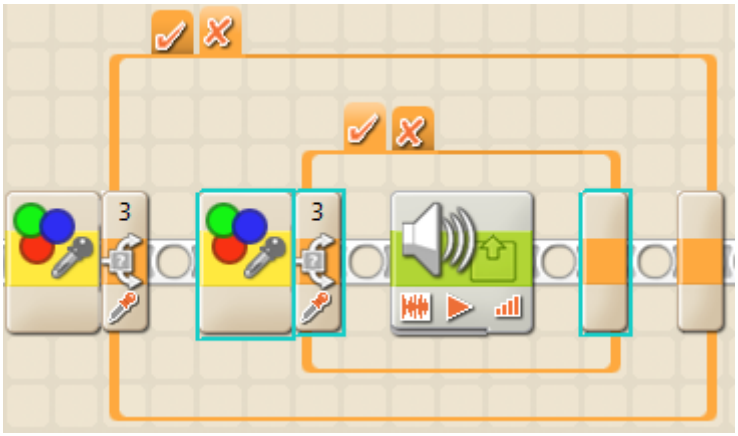
Obrázek 27: Schéma úlohy Sound

[9]Servomotor se dá nastavit v rozmezí od -1 do 1, kde pro zápornou hodnotu jde o zpětný chod.

Na obrázku 27 je vidět službu LegoNXTBrickv2, která jak již bylo dříve zmíněno je potřebná pro komunikaci s NXT kostkou. Další službou je pak služba mic, která slouží ke komunikaci se zvukovým senzorem. Výstup této služby je pak vstupem pro aktivitu Calculate, která ze senzoru získává intenzitu zvuku (proměnná Intensity. Pro přehlednější zpracování a možné pozdější využití, je hodnota uložena to proměnné s názvem intenzita (aktivita Variable). Nyní, když máme nastavenu hodnotu intenzity zvuku okolí, přichází na řadu rozhodovací blok (aktivita If), který určí, jak rychle se budou servomotory otáčet. Spojnice mezi výstupy z rozhodovacího bloku a vstupem služby pro ovládání servomotoru nastavují rychlost otáčení (viz předchozí odstavec).

5.9 Color senzor

Tento program je první z posledních tří, které byly vytvořeny v NXT-G. Jak již bylo zmíněno dříve, VPL nepodporuje všechny použité senzory pro tuto práci, barevný senzor je jedním z nich. Tento senzor dovede rozpoznat celkem šest barev (bílou, černou, červenou, modrou, zelenou a žlutou). Postupným přikládáním kuliček k tomuto senzoru robot vyhodnotí a řekne název barvy. Na výstupu tohoto senzoru je hodnota barvy a podle rozhodovacího bloku pak robot buď řekne její název, nebo pokračuje dále k dalšímu rozhodovacímu bloku. Celý proces se opakuje v nekonečné smyčce, můžeme tedy po spuštění programu neustále přikládat barevné předměty a testovat, zda je robot dobře rozpoznává.

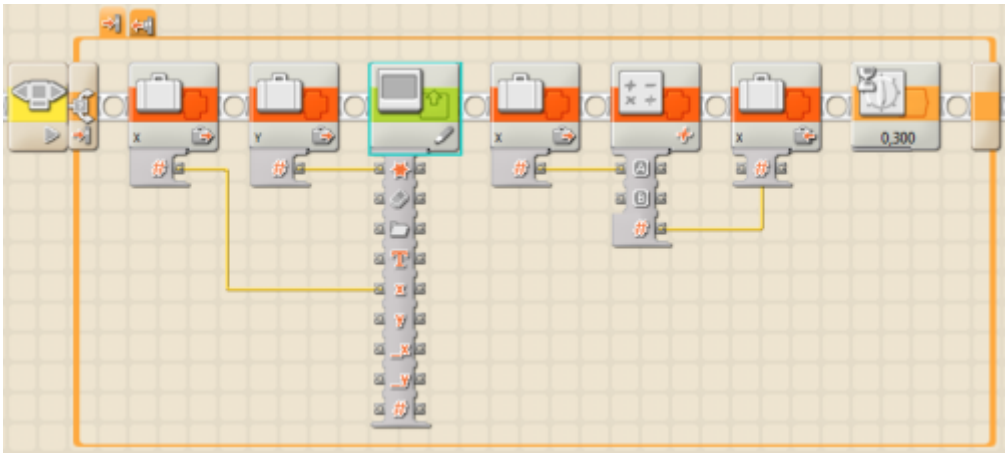


Obrázek 28: Část schéma pro úlohu Color senzor

Na obrázku 28 je zobrazena část programu, poslední dva rozhodovací bloky. Program se skládá z celkem šesti rozhodovacích bloků (podmínek if respektive else-if).

5.10 Drawing

Druhým z programů vytvořených v NXT-G je kreslení. Po spuštění aplikace se vyčistí display na NXT a pomocí šipek a enter tlačítka můžeme začít na display postupně vykreslovat pixely. Šipka do prava respektive do leva vykresluje postupně pixely na display do prava respektive do leva. Stisknutím kombinace tlačítek enter a pravé šipky začnou se vykreslovat pixely od shora dolů. Naopak stisknutím tlačítka enter a levé šipky, vykreslují se pixely od zdola nahoru. Postupně tak můžeme celý dispaly zaplnit vykreslenými pixely. Pro ukončení programu slouží čtvrté tlačítko (esc).



Obrázek 29: Část schéma pro úlohu Drawing

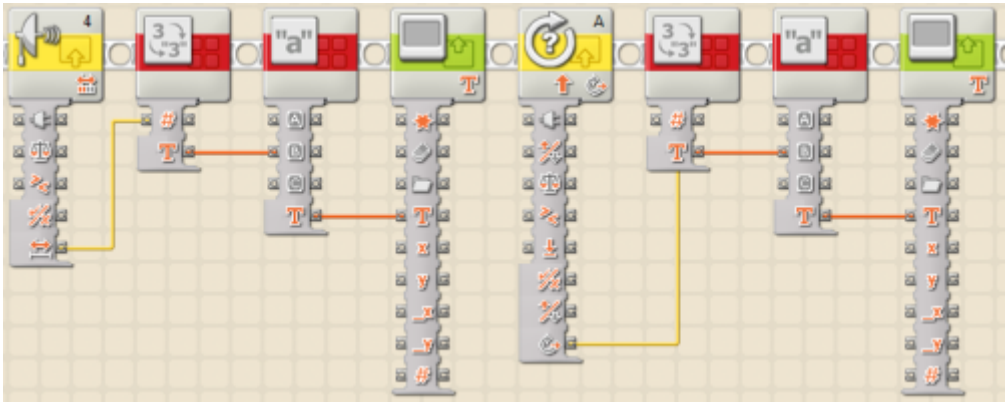
Na obrázku 29 je zobrazena část programu, konkrétně smyčka pro vykreslování pixelů, když je zmáčknuta pravá šipka. Proměnné X a Y slouží jako souřadnice, kde se právě nachází pomyslný kurzor, pomocí kterého se vykreslují pixely. Při každém posunutí kurzoru se provede přepočet a vyčká se 300ms, než se začne opět vykreslovat.

5.11 Info NXT

Poslední, třetí program vytvořený v NXT-G demonstruje výpis informací pro čtyři senzory, tři servomotory a tři tlačítka, které se nacházejí na NXT kostce. Každému senzoru respektive servomotoru respektive tlačítkům náleží jeden řádek. Každý, kromě posledního, na kterém se zobrazují názvy tlačítek, je očíslován a pojmenován podle senzoru respektive servomotoru. Za pojmenováním jsou pak v reálném čase vypisovány hodnoty pro jednotlivé senzory respektive servomotory.

Mezi použité senzory patří: dotykový senzor (první vstup), zvukový senzor (druhý vstup), barevný senzor (třetí vstup) a ultrazvukový senzor (čtvrtý vstup). U jednotlivých servomotorů je měřena a zobrazována ujetá dráha v centimetrech. Této vlastnosti

se dá využít například pro měření rozměrů balíků. Při otáčení servomotoru opačným směrem se vzdálenost snižuje a při překročení nuly se před hodnotou objeví znaménko mínus. Pro výpis jednotlivých hodnot bylo nutné ošetřit jejich úplné přepisování, jinak totiž docházelo, k výpisu chybných hodnot respektive staré byly částečně přepsány novými.



Obrázek 30: Část schéma pro úlohu Info NXT

Na obrázku 30 je zobrazena část programu, konkrétně části pro ultrazvukový senzor a první servomotor. Jak je z obrázku patrné, hodnoty pro senzor respektive servomotor jsou převedeny na textový řetězec, který je následně zpracován textovým blokem a jeho výstup je poslán na display. Pro blok display je nutné nastavit na kterém řádku a sloupci se budou hodnoty zobrazovat, aby nedocházelo k chaotickým výpisům.

6 Závěr

Cílem této diplomové práce bylo popsat použití jazyka VPL, navrhnout, naimplementovat a otestovat ukázkové aplikace pro robota NXT. Postupně bylo představeno prostředí MRDS, jakou má architekturu a jaké prostředí do něj patří. Celá jedna kapitola byla pak věnována prostředí VPL, na které je tato práce zaměřena. Zjistili jsme, že jej nemusí využívat jen zkušení programátoři, ale i začátečníci, kteří si chtějí vyzkoušet, jaké to je si něco naprogramovat i když trošku netradiční formou. Vizuální programování je sice zábavné a pro menší aplikace není obzvlášť složité, ale přináší řadu omezení, která se pak musí řešit překompilováním vizuálního kódu do zdrojového a jeho další vnitřní úpravou.

K otestování vytvářených aplikací byl použit robot LEGO MINDSTORMS NXT ve verzi 2.0. Spolu s tímto robotem je dodáváno i vývojové prostředí NXT-G, ve kterém se rovněž programuje vizuálně. Toto prostředí ovšem oproti VPL nabízí mnohem větší možnosti v nastavování jednotlivých služeb a funkcí. NXT-G je vyvíjeno přímo konkrétně jen pro NXT roboty, oproti tomu VPL je zaměřeno na různé druhy robotů a tak strádá v konkrétnějším využití. U tři navržených programů, pro tuto práci, tak muselo být zkombinováno prostředí VPL s prostředím NXT-G z důvodů otestování všech dostupných senzorů. Celkem, pro tuto práci, bylo navrženo a popsáno deset aplikací (dílčích úloh).

MRDS bylo nainstalováno pod operačním systémem Windows Vista, později pak pod Windows 7. Ukázkové programy, které jsou dostupné, ať už po nainstalování MRDS nebo na internetových stránkách MSDN, jsou vcelku stručné a výstižné, ale nijak rozsáhlé, což je dáno i malou pamětí robota. Pro názornou ukázku a uvedení do problematiky mobilních robotů je tato stavebnice dobrou a cenově přijatelnou volbou.

Pojem vizuální programování pro mne nebyl neznámý, ale nikdy jsem se do této doby neměl příležitost setkat. Jedná se o příjemnou formu programování, kde i úplný začátečník dokáže vytvořit jednoduchý program a postupně tak získávat zkušenosti s touto oblastí. Jelikož jsem měl možnost vyzkoušet si jak prostředí VPL, tak prostředí NXT-G nabízí se otázka, které z nich je lepší. Nemusím ani na chvíli váhat, abych řekl, že určitě NXT-G, které je mnohem propracovanější po vizuální stránce programování. Ale vývoj jde neustále dopředu a nová verze VPL (celého MRDS) bude mít zase větší možnosti využití a naprogramování nejen pro NXT roboty.

7 Reference

[1] Ladislav Kopecký. *Porovnání prostředí pro řízení mobilních robotů [online].* <http://cyber.felk.cvut.cz/research/theses/papers/84.pdf> [cit. 2009-06-20]

[2] nxtprograms.com. *Internetové stránky s množstvím aplikací pro Lego Mindstorms NXT roboty [online].* <http://www.nxtprograms.com/> [cit. 2010-03-02]

[3] Lego Mindstorms NXT. *Internetové stránky popisující problematiku Lego Mindstorms NXT v MRDS [online].* <http://msdn.microsoft.com/en-us/library/dd939240.aspx> [cit. 2010-03-02]

[4] ProMRDS. *Internetové stránky věnující se knize Professional Microsoft Robotics Developer Studio [online].* <http://www.promrds.com/> [cit. 2010-04-20]

[5] Software LEGO® MINDSTORMS® Education NXT. *Český překlad manuálu.* <http://www.eduxe.cz/download/index.htm> [cit. 2009-04-22]

[6] O čem sní roboti. *Recenze Lego Mindstorms NXT [online].* <http://www.zive.cz/clanky/o-cem-sni-roboti-recenze-lego-mindstorms-nxt/sc-3-a-133790/default.aspx> [cit. 2007-01-16]

[7] LEGO Mindstorms NXT 2.0, 8547, 8547, Robotics Invention System, Info & Sale Prices *Internetový obchod s produkty Lego Mindstorms NXT 2.0 [online].* <http://www.mrrobot.com/lego/> [cit. 2010-04-06]

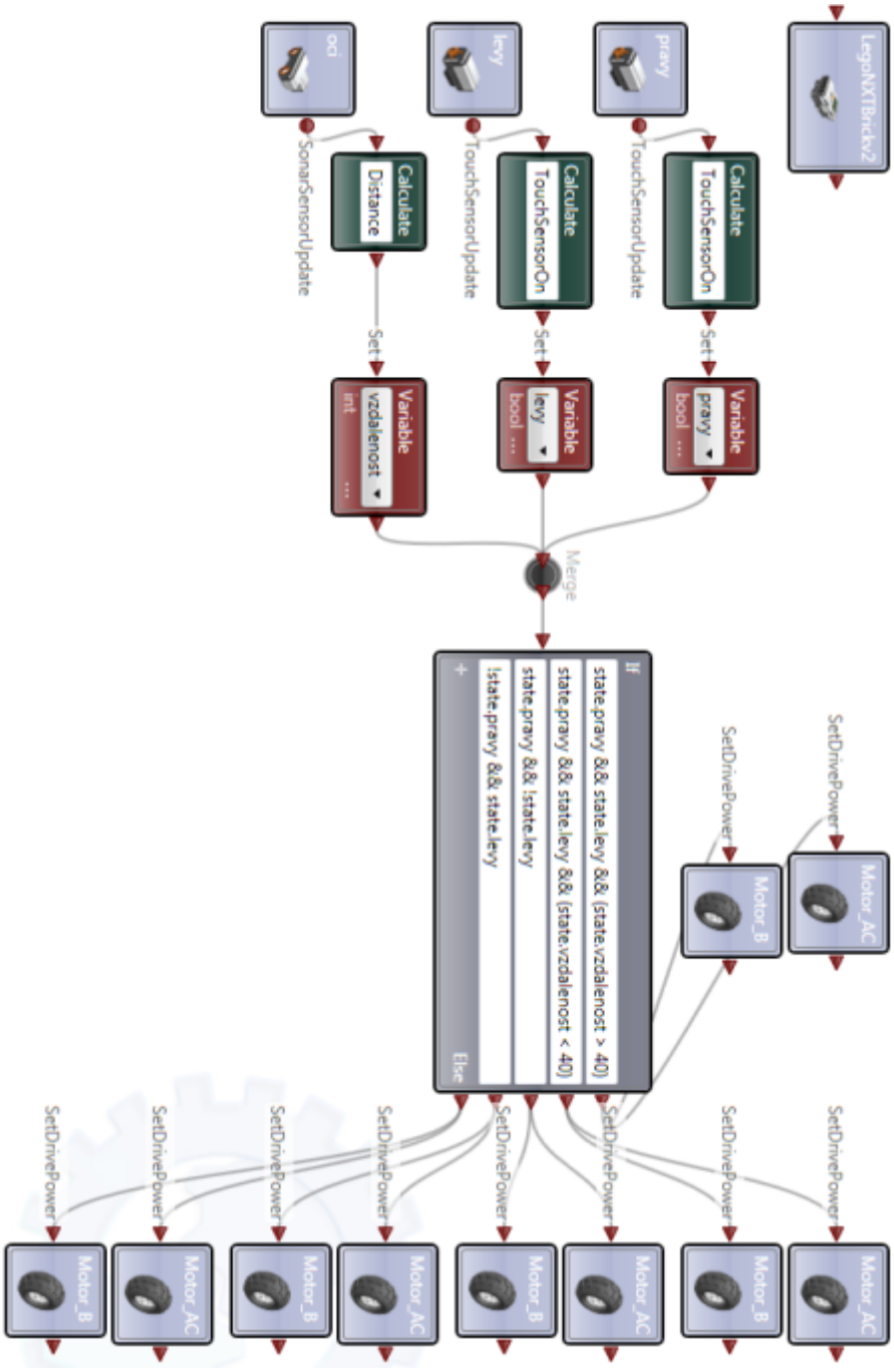
[8] LDraw.org :: The Central Site for the LDraw Family of LEGO® CAD Software *Internetový portál zabývající se Lego produkty [online].* <http://www.ldraw.org/> [cit. 2010-04-06]

[9] RobotShop - Personal and Professional Robots, Robot Parts, Robot Kits, Robot Repair. *Internetový obchod věnující se osobní a profesionální robot technologii* <http://www.robotshop.ca/> [cit. 2010-04-06]

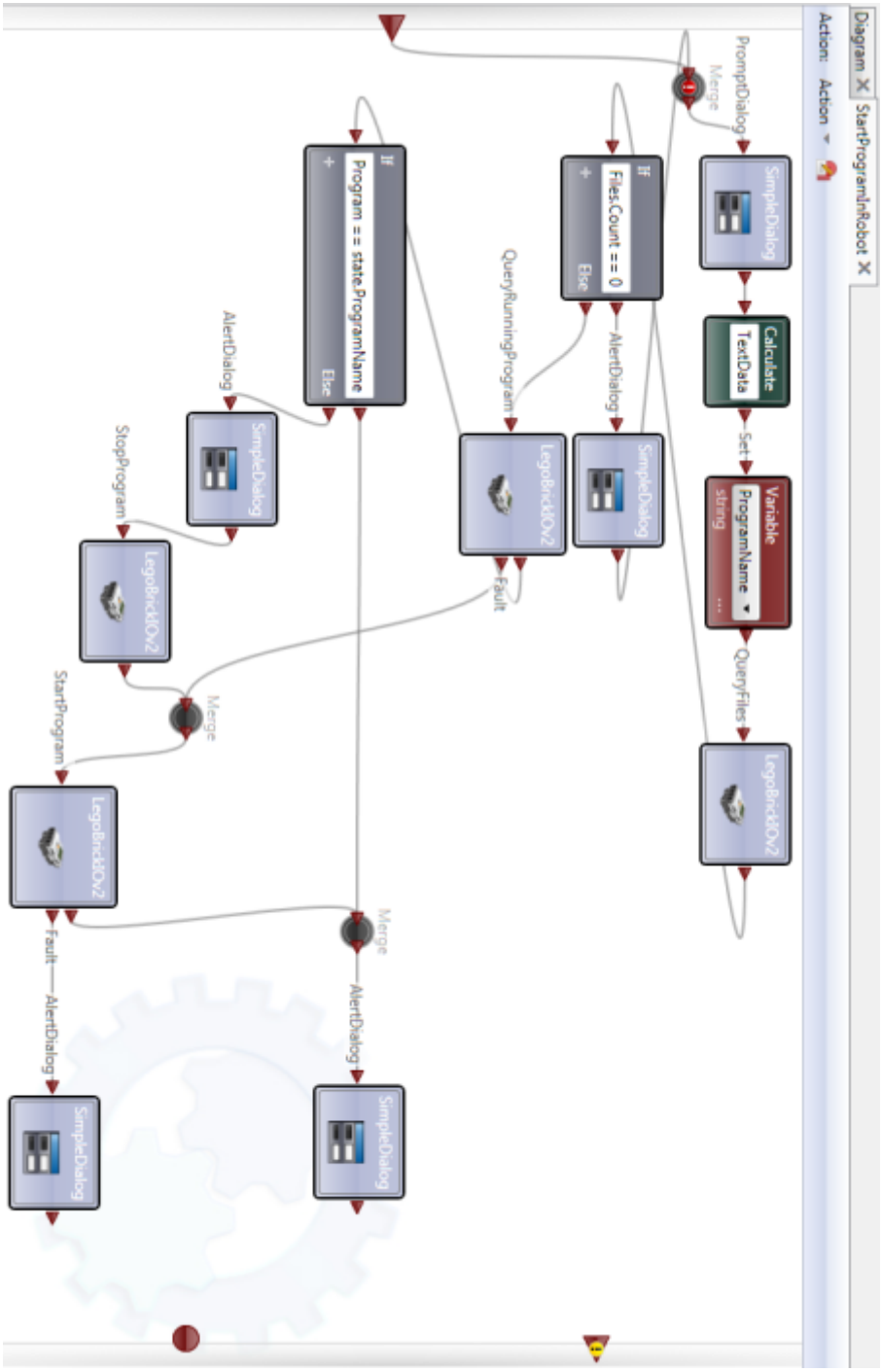
[10] Port - Wikipedie, otevřená encyklopedie, *Internetová encyklopedie [online].* <http://cs.wikipedia.org/wiki/Port> [cit. 2010-04-29]

[11] Extra-torrent - PC - Fyzika podle ATi Ageia PhysX killer? *Internetová stránka věnující se problematice AGEIA PhysyX [online].* <http://www.extra-torrent.estranky.cz/clanky/pc-/fyzika-podle-ati--ageia-physx-killer_> [cit. 2007-02-07]

A Obrázek k úloze Shooter a Start program



Obrázek 31: Schéma úlohy Shooter



Obrázek 32: Schéma aktivity StartProgramInRobot (jsou zde vidět i záložky s diagramy)

B Obsah CD

- **instal** - složka s prostředím MRDS, ve kterém bylo vytvořeno prvních sedm dílčíh úloh, zbylé tři byly vytvořeny v NXT-G, který je dodáván spolu s robotem
- **text** - složka s dokumentem diplomové práce v elektronické podobě (formát pdf)
- **úlohy** - složka s jednotlivými úlohy pro robota (úloha Lini Follower obsahuje složku C#, kde je úloha z VPL překompilována do C# kódu)